A PROJECT
REPORT ON

HOSPITAL MANAGEMENT SYSTEM

By

Krunal Kishnadwala (CE209)
(20CEUBD013)
Jash Tamakuwala (CE224)
(20CEUBD006)

B.Tech CE Semester - IV

Subject: Software Engineering Principles and Practices

Guided by:
Prof. Pinkal Chauhan
Prof. Ankit Vaishnav

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University

Dharamsinh Desai University, Nadiad

Faculty Of Technology
Department Of Computer Engineering

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of softwareengineering principles and practice and recorded in this journal is the bonafied work of

Krunal Kishnadwala (CE - 209) (20CEUBD013)
Jash Tamakuwala (CE - 224) (20CEUBD006)

Of B.tech semester IV in the branch of computer engineering during

the Academic year 2021.

Prof.Jigar M. Pandya
Assistant Professor
Dept. Of Computer Engg,
Faculty Of Technology
Dharamsinh Desai University,Nadiad

Dr. C. K. Bhensdadia
Head of Department
Dept. Of Computer Engg,
Faculty Of Technology,
Dharamsinh Desai University
Nadiad

Abstract

Hospital Management System is the process in which the details of patients as well as doctors are maintained. It’s like a website where a patient/doctor have to register into it to get its benefits.One patient/doctor have registered they will get an email id and password so that all the data is secure.If patient then patient can request for appointment, get the reports rather than going to clinic or hospital to collect it.Even the doctors will register to render their services to the people.They accept or reject the appointment they receive, also they will add the reports of their patients.

Introduction

Hospital Management System in which there are mainly three users Admin,Doctors and Patients.Patients can book appointments according to disease.Admin can modify doctors and patients details.Patients and doctors can login and watch scheduled appointments.Doctors can treat patients accordingly.

Tools/Technologies Used

Technologies :

- Django
- Python
- MySQL
- Bootstrap
- JavaScript
- HTML

Tools :

- Git
- PyCharm

Software Requirement Specifications

Functional requirements:

Hospital Management System(HOMS):

R 1: Manage Patients information

Description: Users can add/modify/delete details of Patients.

R 1.1: **Create account of patient**
Description: User can add details of patients.
Input: Details of patients.
Output: Account created with patient's id.

R 1.2: **Update details of patients**
Description: User can update details of patients.
Input: Patients id/Name and change needed to be made.
Output: Details updated.

R 1.3: **Delete account**
Description: Patients can delete account.
Input: Patients id/Name
Output: Account deleted.

R 1.4: **Display details of account**
Description: Patients can view account.
Input: Patients id/Name
Output: Patients details

R 2: Manage appointments

Description: user can add/modify/delete the appointments.

R 2.1: Create appointments

Description: Users can add details to create appointments.
Input: Details of patients.
Output: Appointment generated with appointment number.

R 2.2: Update appointments

Description: User can update details of appointment.
Input: Appointment number
Output: Appointment updated.

R 2.3: Delete appointments

Description: Users can delete appointment.
Input: Appointment number
Output: Appointment deleted.

R 2.4: Display details of appointments

Description: User can view appointments.
Input: User id/Name
Output: Appointments details

R 3: Manage medical staff information

Description:Admin can add/modify/delete details of medical staff like doctors,nurse,wardboy etc.

R 3.1: Create account of staff

Description: User can add details of staff.
Input: Details of staff.

Output: Account created with staff id.

R 3.2: **Update details of staff**
Description: User can update details of staff.
Input: Staff id/Name and change needed to be made.
Output: Details updated.

R 3.3: **Delete appointments**
Description: User can delete account.
Input: Staff id/Name
Output: Account deleted.

R 3.4: **Display details of appointments**
Description: User can view appointments.
Input: User id/Name
Output: Appointments details

R 4: Generate Medical Reports/Prescriptions

Description: Doctors can generate medical reports and prescriptions for the patients.

R 4.1: **Create medical report/prescriptions**
Description: Users can create medical report/prescriptions.
Input: Details of medical report/prescriptions.
Output: Medical report/prescriptions created.

R 4.2: **Update details medical report/prescriptions**
Description: Users can update details of medical reports/prescriptions.
Input: change needed to be made.
Output: Details updated.

R 4.3: **Display medical report/prescriptions.**
Description: Users can view medical report/prescriptions.
Input: User id/Name.
Output: Medical report/prescriptions.

R 4.4: **Download medical report/prescriptions.**
Description: Users can download medical reports/prescriptions.
Input: User id/Name.
Output: Downloaded Medical report/prescriptions.

R 5:Manage Accounting

Description:User can generate invoice and take payments from the patients.

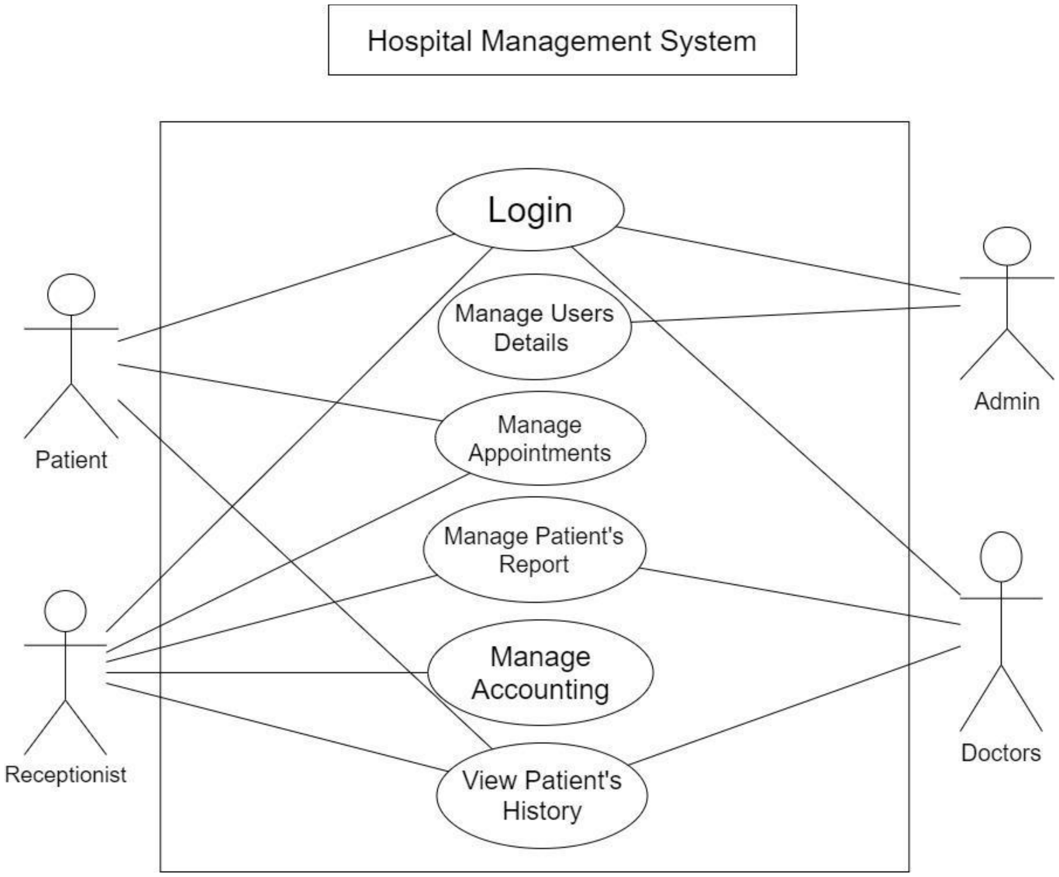
R 5.1: Make payment.
Description: Users can make payment through online gateways.
Input: User id/Name.
Output: Generated invoice.

R 5.2: Create Invoice.
Description: User can generate an invoice.
Input: User id/Name.
Output: Display invoice.

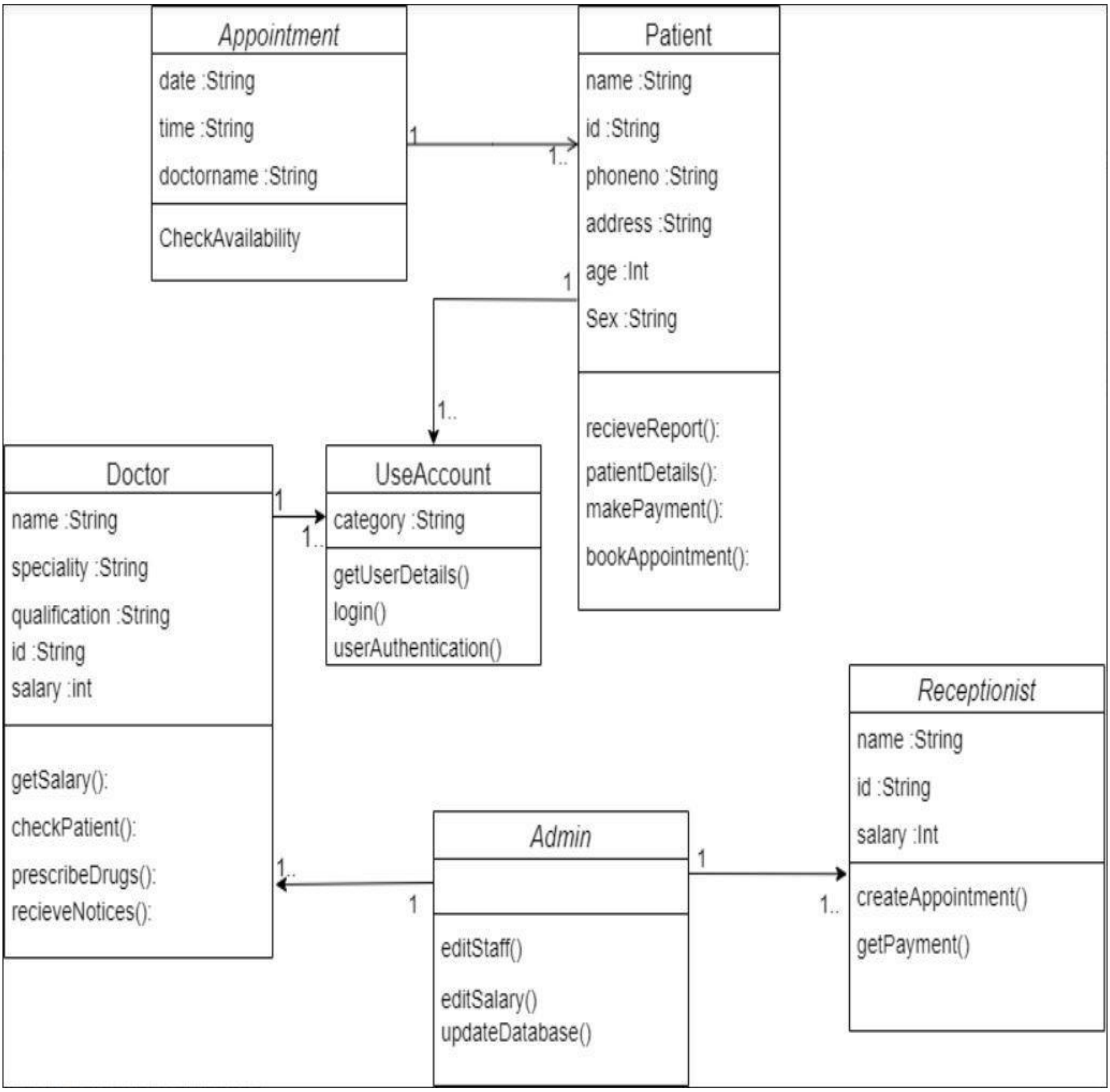
R 5.3: Download Invoice.
Description: User can download invoice.
Input: User id/Name.
Output: Downloaded invoice.

Design

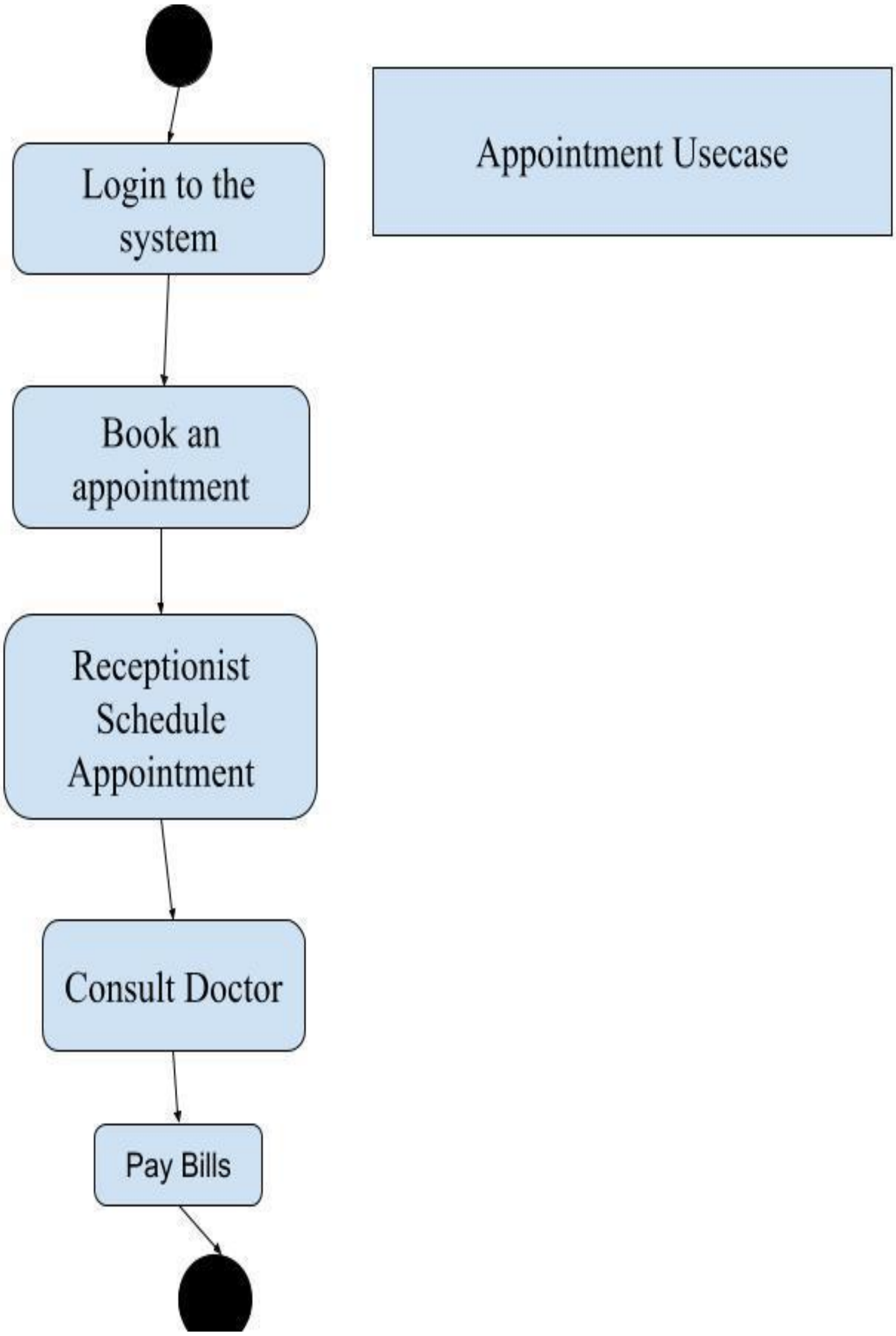
Use case diagram



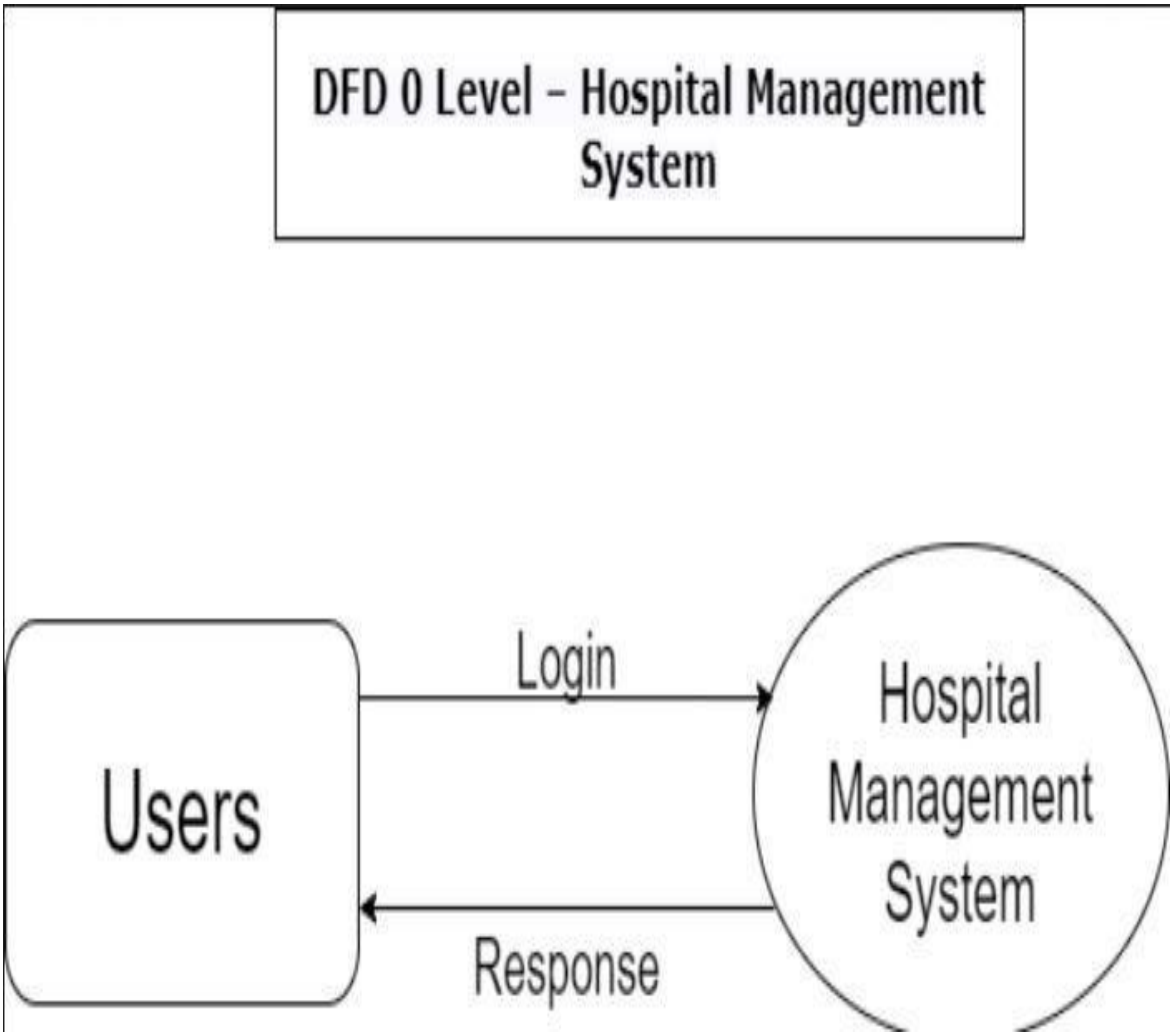
Class Diagram

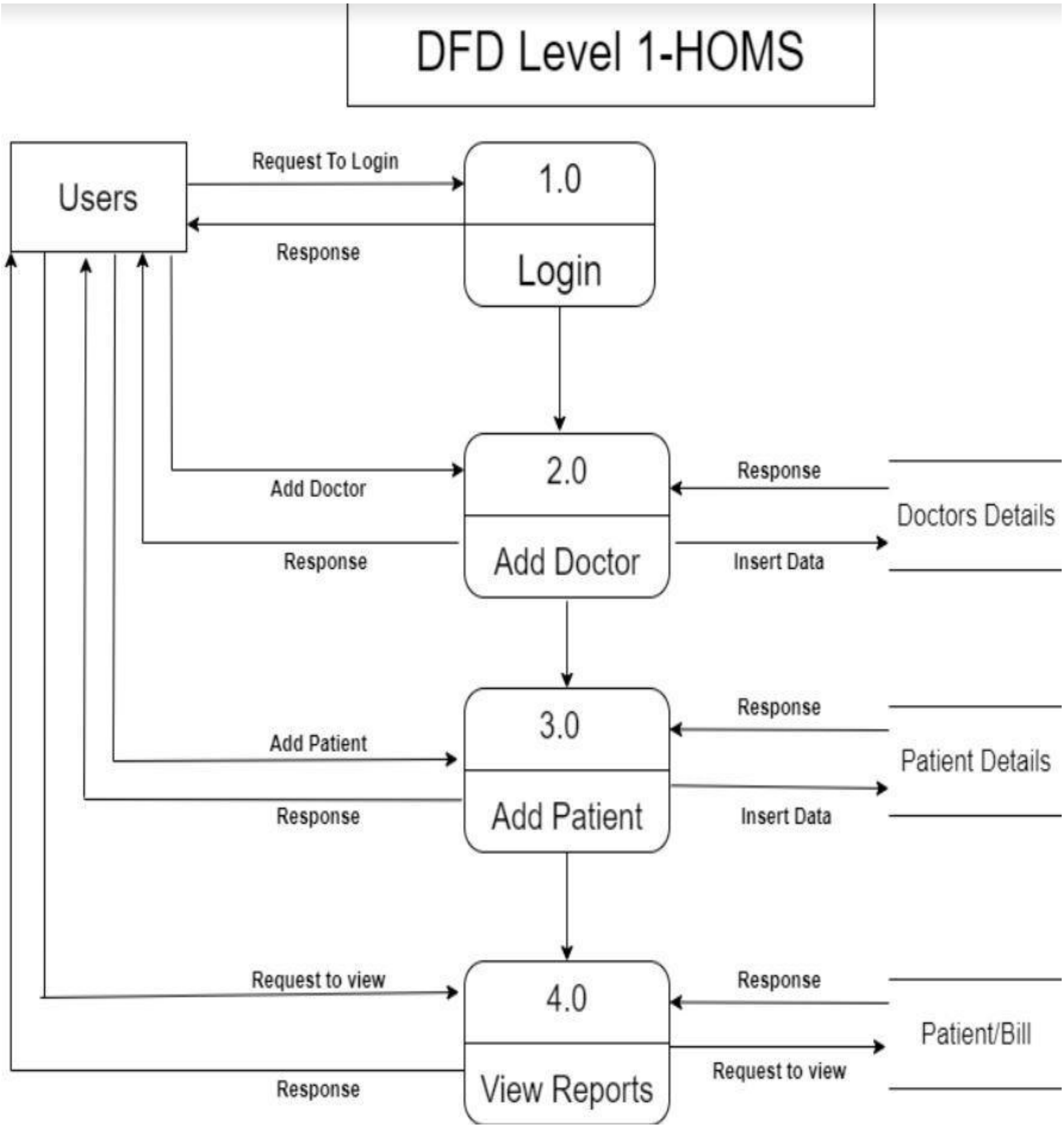


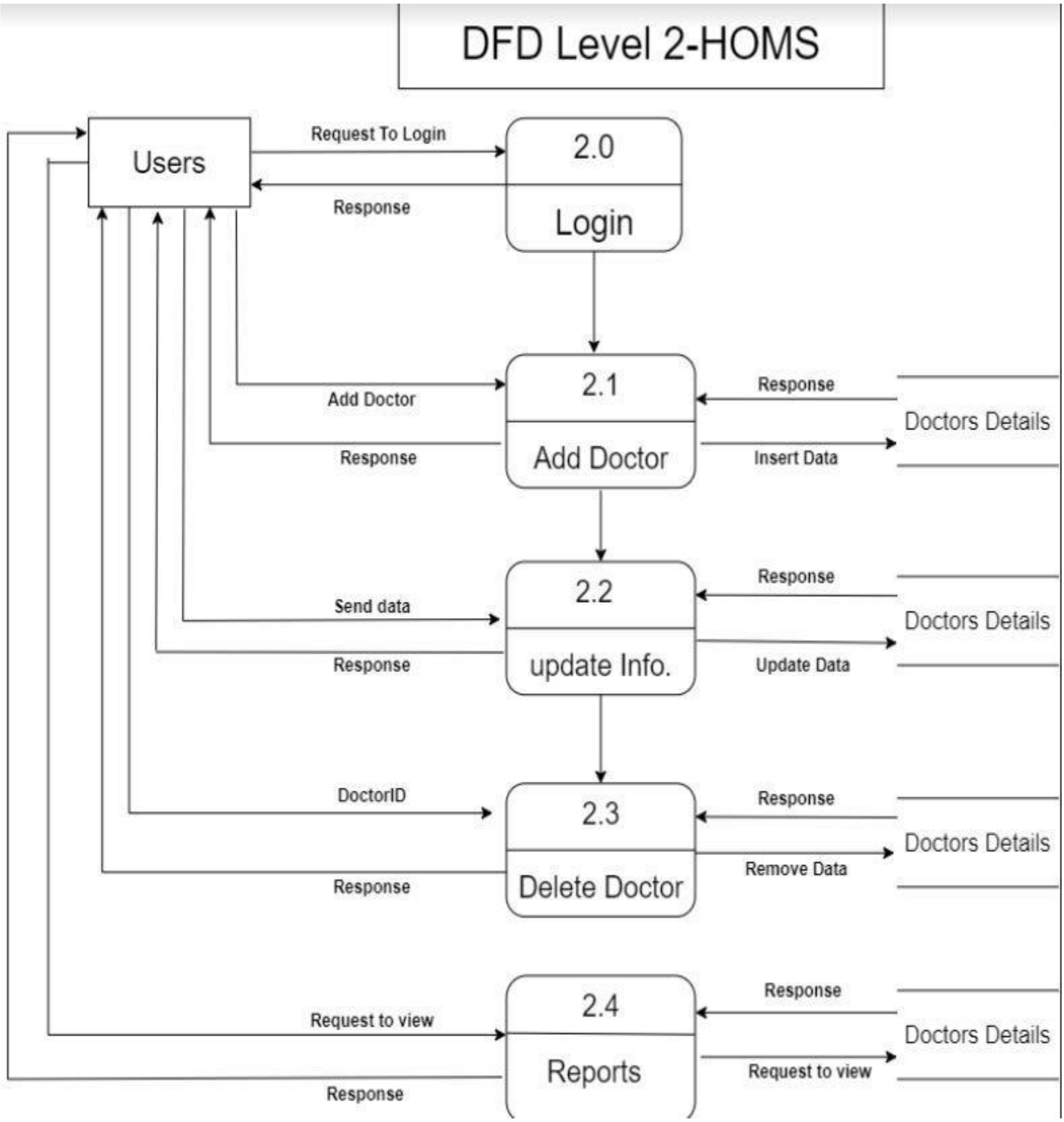
Activity Diagram

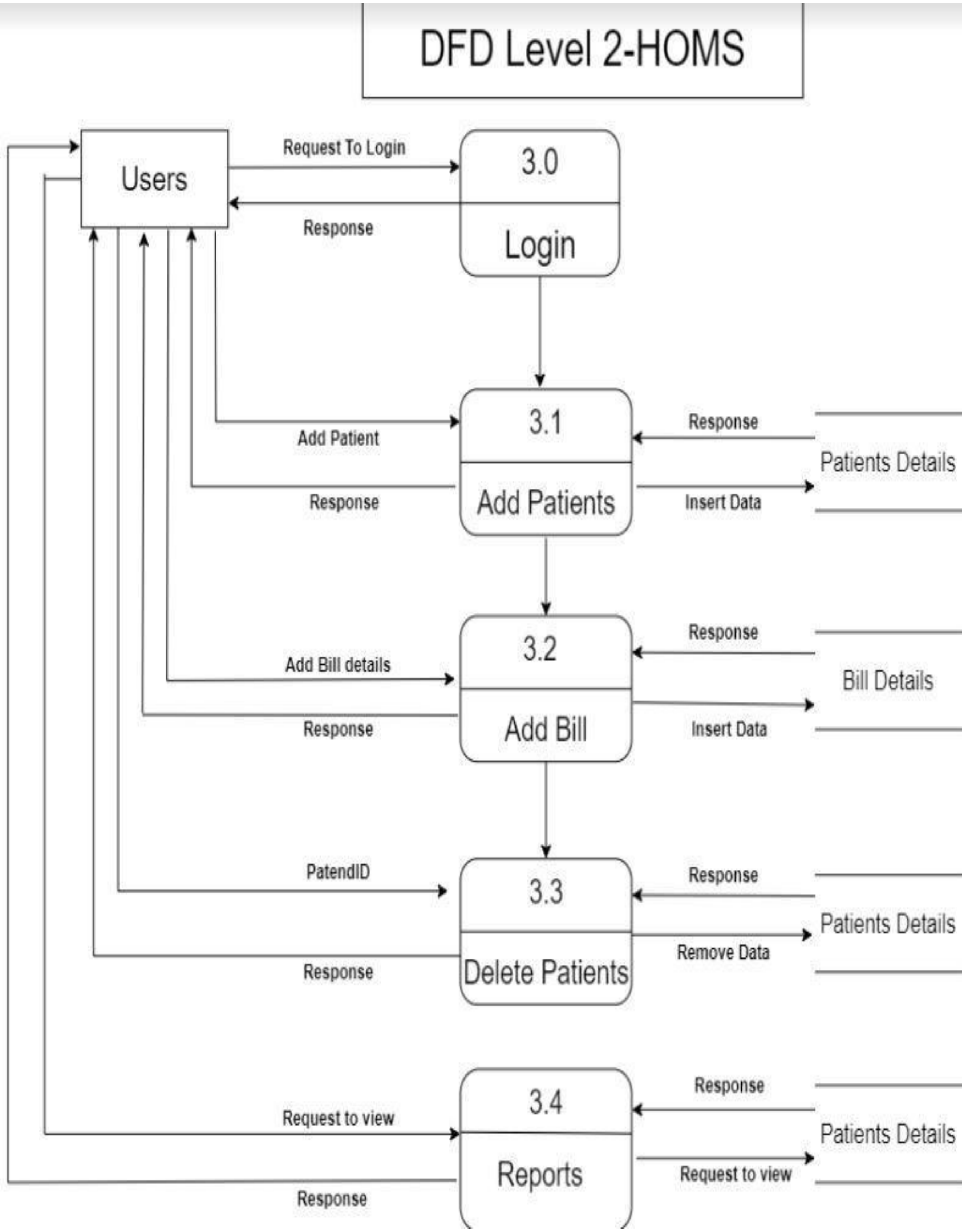


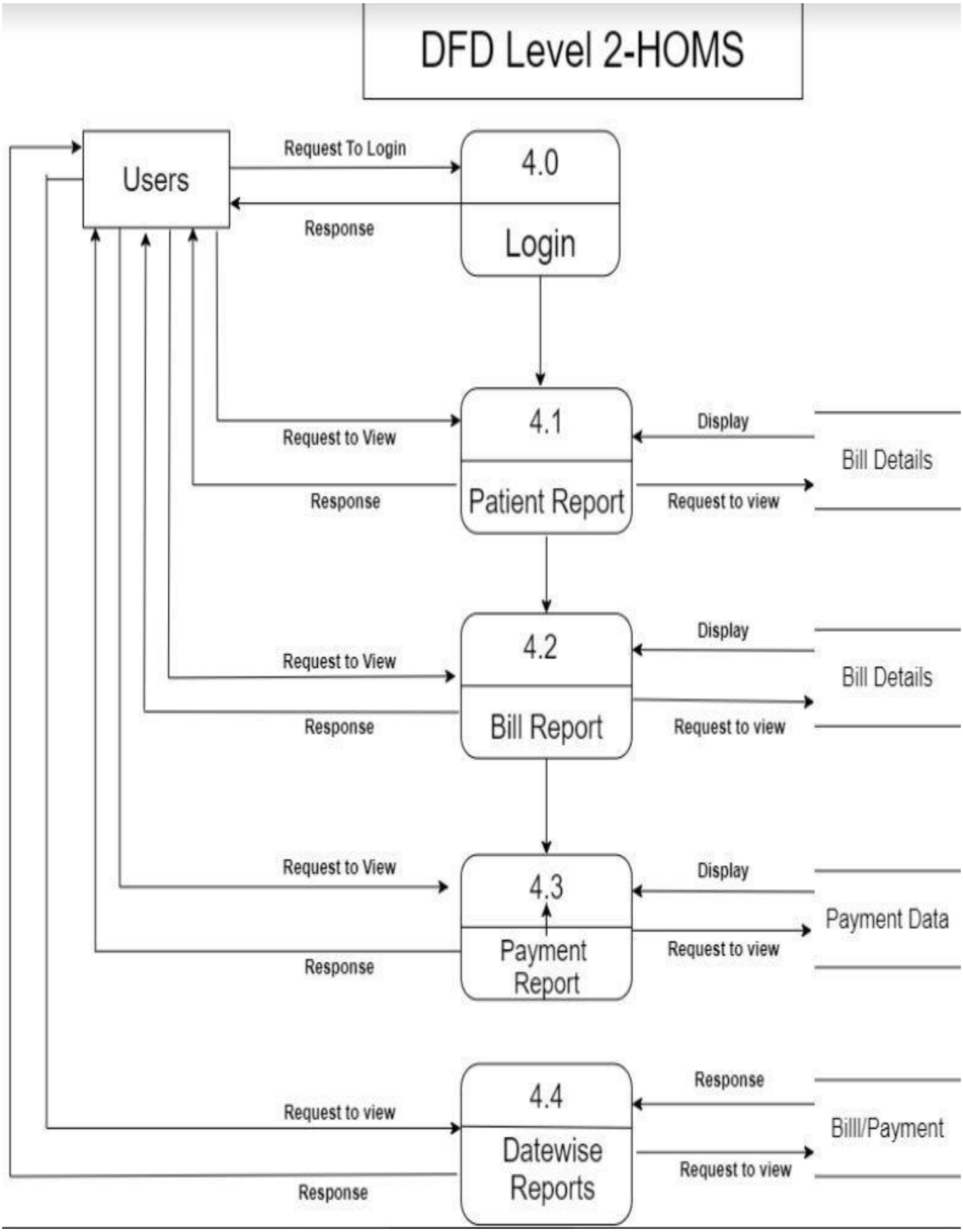
Data Flow Diagrams

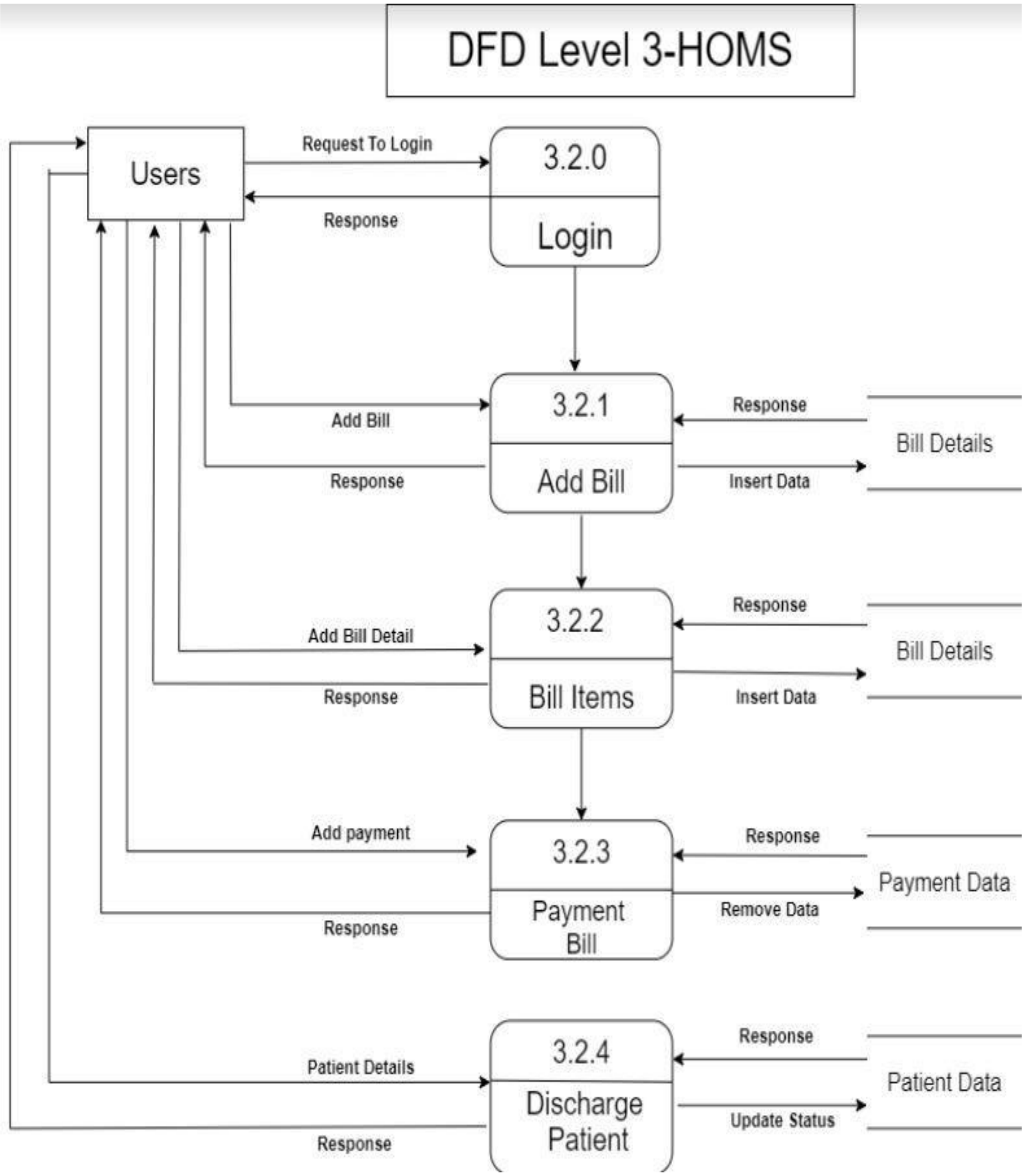

DFD 0 Level – Hospital Management System
Users
Login
Response
Hospital Management System


DFD Level 1-HOMS
Users
Request To Login
Response
1.0
Login
2.0
Add Doctor
Add Doctor
Response
Response
Insert Data
Doctors Details
3.0
Add Patient
Add Patient
Response
Response
Insert Data
Patient Details
4.0
View Reports
Request to view
Response
Response
Request to view
Patient/Bill

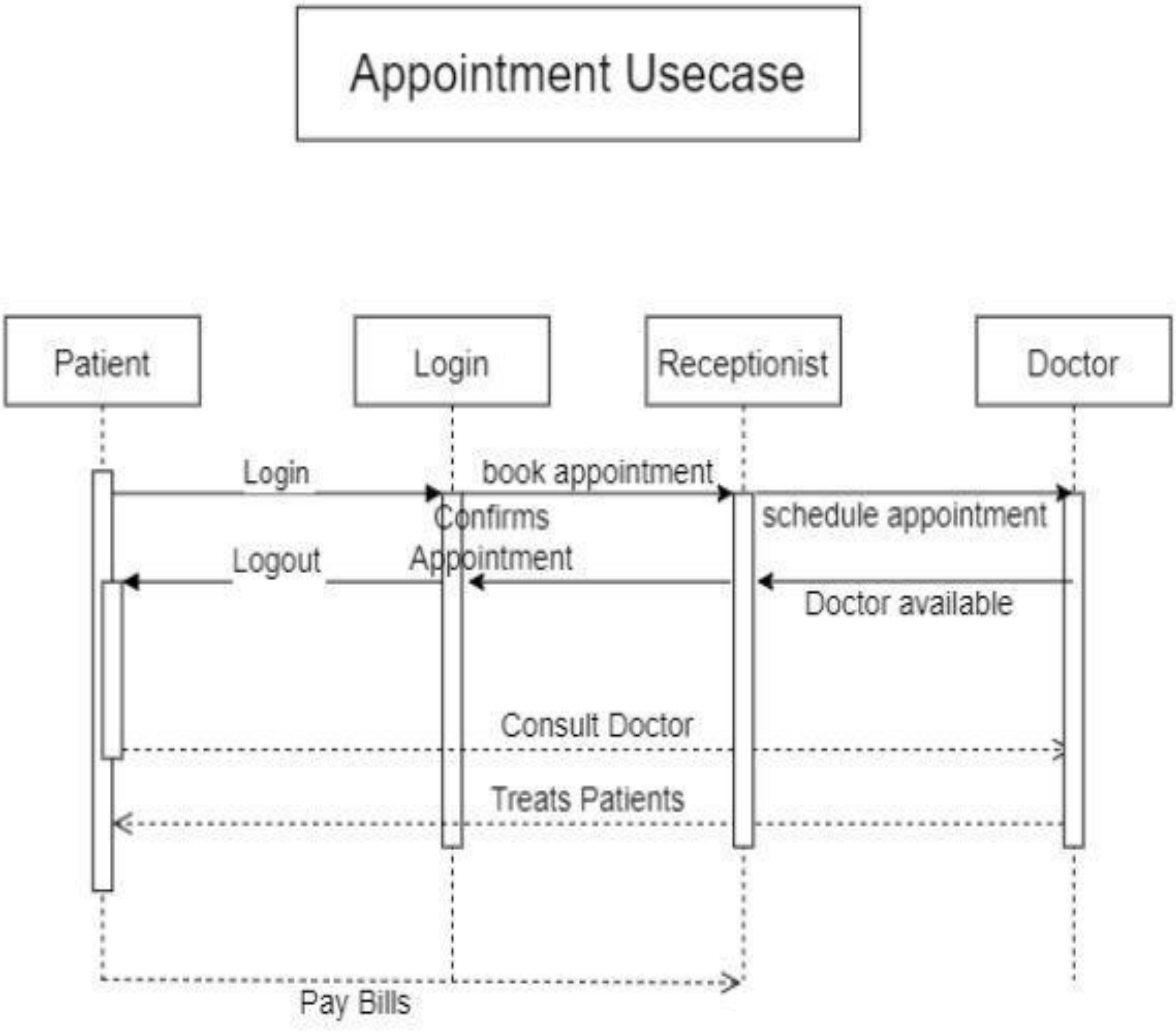

DFD Level 2-HOMS
Users
Request To Login
Response
2.0
Login
2.1
Add Doctor
Add Doctor
Response
Response
Insert Data
Doctors Details
2.2
update Info.
Send data
Response
Response
Update Data
Doctors Details
2.3
Delete Doctor
DoctorID
Response
Response
Remove Data
Doctors Details
2.4
Reports
Request to view
Response
Response
Request to view
Doctors Details


DFD Level 2-HOMS
Users
Request To Login
Response
3.0
Login
3.1
Add Patients
Add Patient
Response
Response
Insert Data
Patients Details
3.2
Add Bill
Add Bill details
Response
Response
Insert Data
Bill Details
3.3
Delete Patients
PatendID
Response
Response
Remove Data
Patients Details
3.4
Reports
Request to view
Response
Response
Request to view
Patients Details


DFD Level 2-HOMS
Users
Request To Login
Response
4.0
Login
4.1
Patient Report
Request to View
Response
Display
Request to view
Bill Details
4.2
Bill Report
Request to View
Response
Display
Request to view
Bill Details
4.3
Payment Report
Request to View
Response
Display
Request to view
Payment Data
4.4
Datewise Reports
Request to view
Response
Response
Request to view
Billl/Payment


DFD Level 3-HOMS
Users
Request To Login
Response
3.2.0
Login
3.2.1
Add Bill
Add Bill
Response
Response
Insert Data
Bill Details
3.2.2
Bill Items
Add Bill Detail
Response
Response
Insert Data
Bill Details
3.2.3
Payment Bill
Add payment
Response
Response
Remove Data
Payment Data
3.2.4
Discharge Patient
Patient Details
Response
Response
Update Status
Patient Data

Sequence Diagram



Implements Details

Modules

Admin

- Sign Up their account. Then Login.
- Can register/view/delete doctor
- Can admit/view patient
- Can view/book

Doctor

- Login.
- Can only view their patient details (symptoms, name, mobile) assigned to that doctor by admin.
- Can view their Appointments, booked by admin.
- Can delete their Appointment.

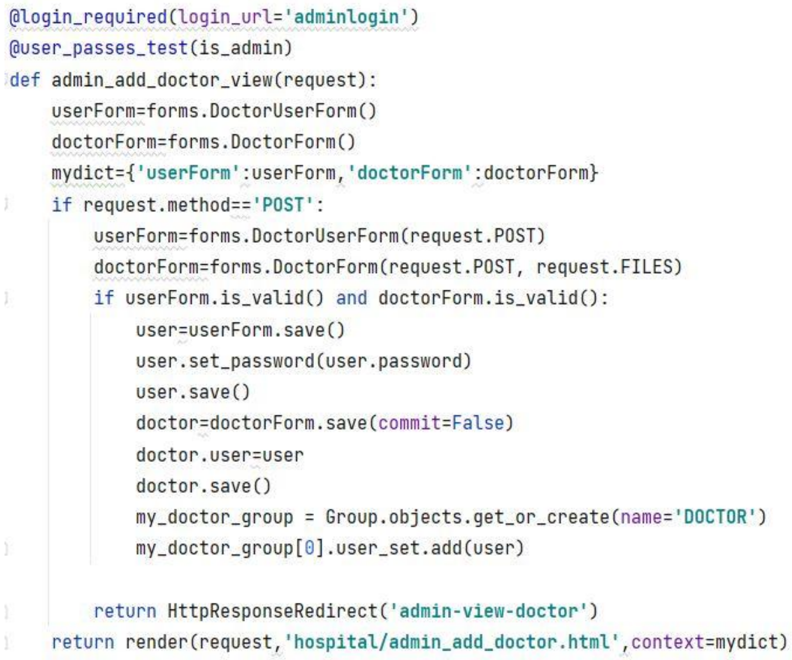
Patient

- Create an account for admission in hospital. Then Login.
- Can view assigned doctor's details.
- Can view their booked appointment.
- Can book appointments.

Major Function Prototypes (Admin)

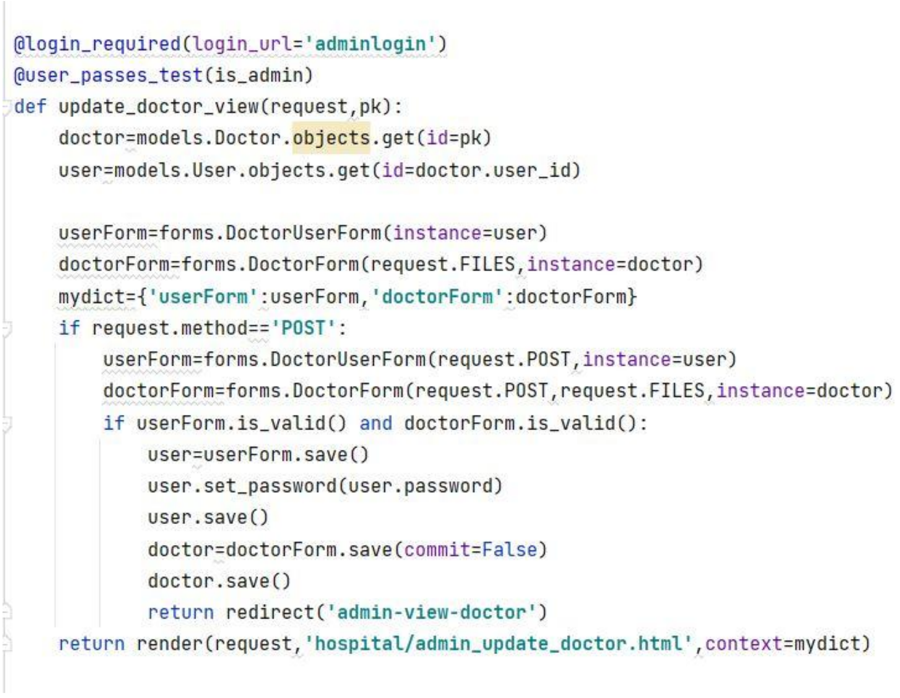
Manage Doctor

Insert Doctor



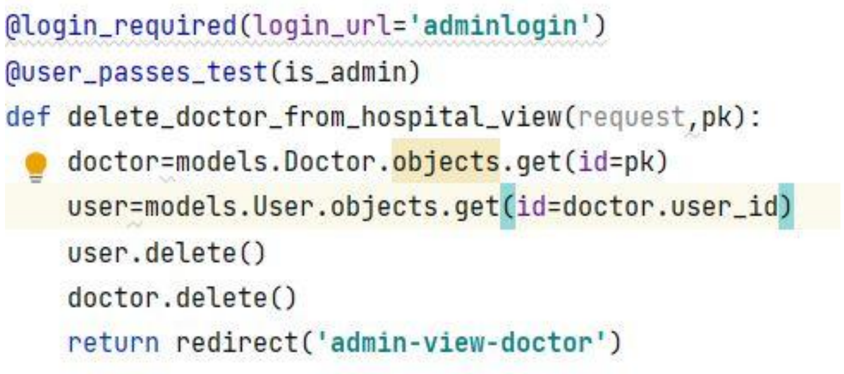
```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_add_doctor_view(request):
    userForm=forms.DoctorUserForm()
    doctorForm=forms.DoctorForm()
    mydict={'userForm':userForm,'doctorForm':doctorForm}
    if request.method=='POST':
        userForm=forms.DoctorUserForm(request.POST)
        doctorForm=forms.DoctorForm(request.POST, request.FILES)
        if userForm.is_valid() and doctorForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()
            doctor=doctorForm.save(commit=False)
            doctor.user=user
            doctor.save()
            my_doctor_group = Group.objects.get_or_create(name='DOCTOR')
            my_doctor_group[0].user_set.add(user)

        return HttpResponseRedirect('admin-view-doctor')
    return render(request,'hospital/admin_add_doctor.html',context=mydict)
```

Update Doctor



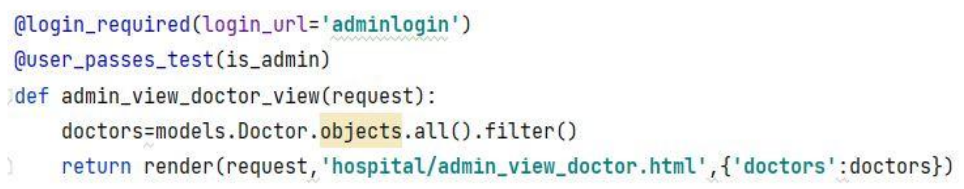
```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def update_doctor_view(request,pk):
    doctor=models.Doctor.objects.get(id=pk)
    user=models.User.objects.get(id=doctor.user_id)

    userForm=forms.DoctorUserForm(instance=user)
    doctorForm=forms.DoctorForm(request.FILES,instance=doctor)
    mydict={'userForm':userForm,'doctorForm':doctorForm}
    if request.method=='POST':
        userForm=forms.DoctorUserForm(request.POST,instance=user)
        doctorForm=forms.DoctorForm(request.POST,request.FILES,instance=doctor)
        if userForm.is_valid() and doctorForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()
            doctor=doctorForm.save(commit=False)
            doctor.save()
            return redirect('admin-view-doctor')
    return render(request,'hospital/admin_update_doctor.html',context=mydict)
```

Delete Doctor



```python
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_doctor_from_hospital_view(request,pk):
    doctor=models.Doctor.objects.get(id=pk)
    user=models.User.objects.get(id=doctor.user_id)
    user.delete()
    doctor.delete()
    return redirect('admin-view-doctor')
```

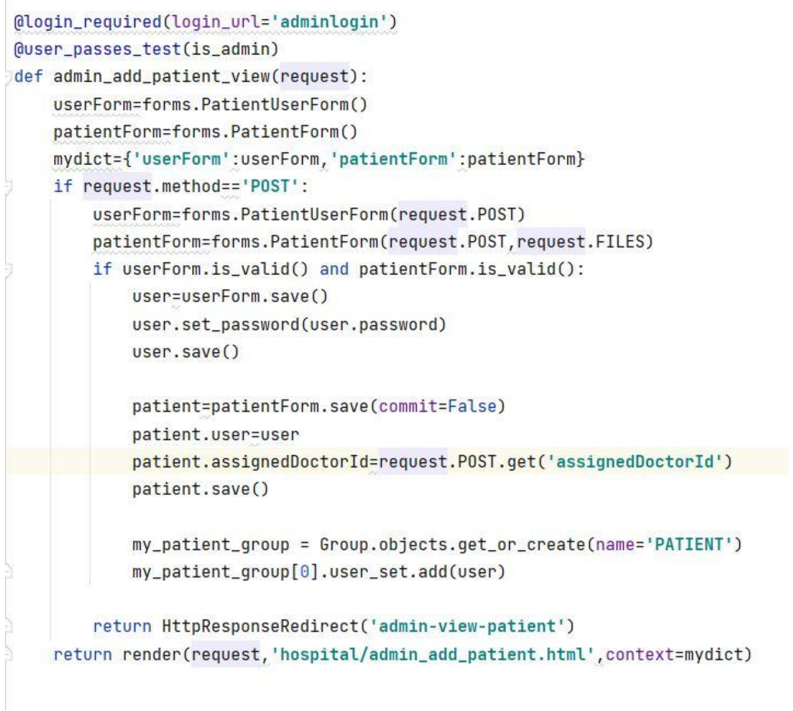
View Doctor



```python
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_doctor_view(request):
    doctors=models.Doctor.objects.all().filter()
    return render(request,'hospital/admin_view_doctor.html',{'doctors':doctors})
```

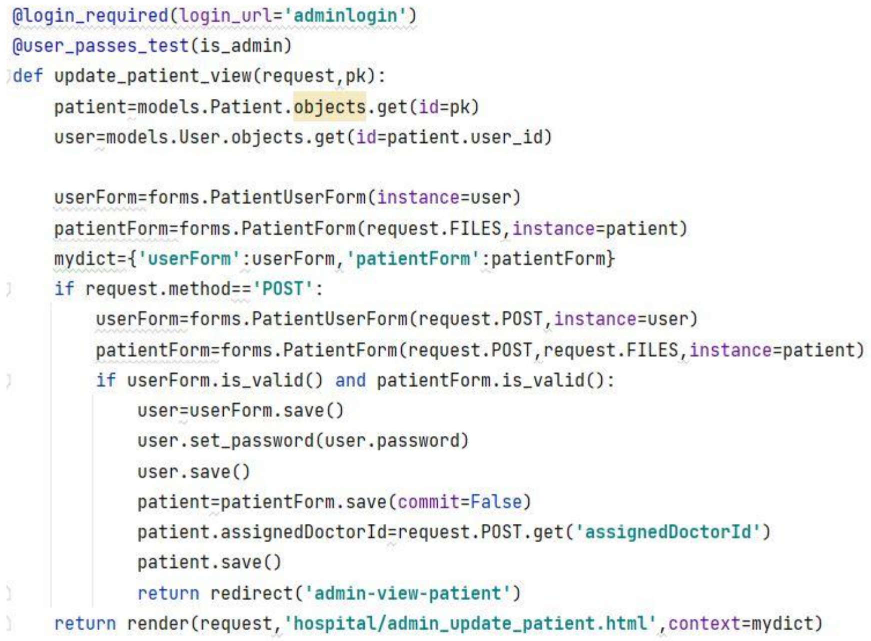
1.Manages Patients

Insert Patients



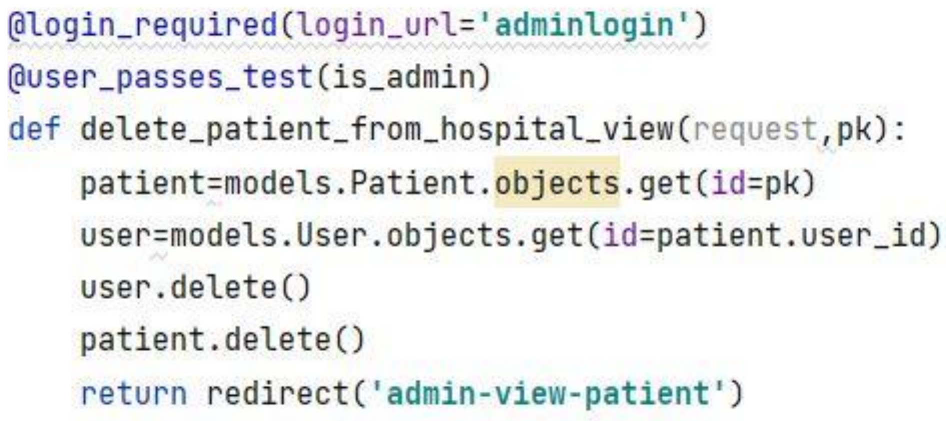
```python
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_add_patient_view(request):
    userForm=forms.PatientUserForm()
    patientForm=forms.PatientForm()
    mydict={'userForm':userForm,'patientForm':patientForm}
    if request.method=='POST':
        userForm=forms.PatientUserForm(request.POST)
        patientForm=forms.PatientForm(request.POST,request.FILES)
        if userForm.is_valid() and patientForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()

            patient=patientForm.save(commit=False)
            patient.user=user
            patient.assignedDoctorId=request.POST.get('assignedDoctorId')
            patient.save()

            my_patient_group = Group.objects.get_or_create(name='PATIENT')
            my_patient_group[0].user_set.add(user)

        return HttpResponseRedirect('admin-view-patient')
    return render(request,'hospital/admin_add_patient.html',context=mydict)
```

Update Patients



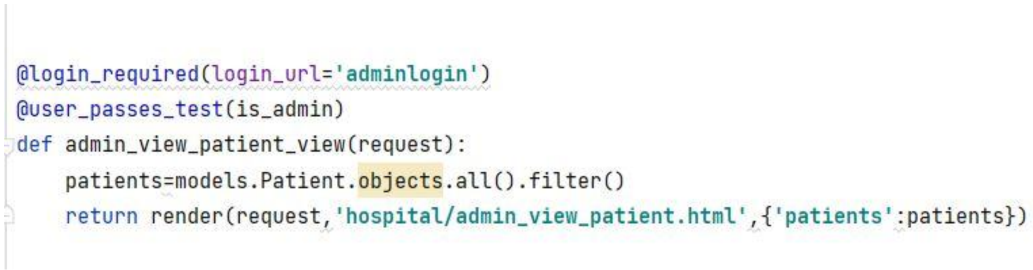
```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def update_patient_view(request,pk):
    patient=models.Patient.objects.get(id=pk)
    user=models.User.objects.get(id=patient.user_id)

    userForm=forms.PatientUserForm(instance=user)
    patientForm=forms.PatientForm(request.FILES,instance=patient)
    mydict={'userForm':userForm,'patientForm':patientForm}
    if request.method=='POST':
        userForm=forms.PatientUserForm(request.POST,instance=user)
        patientForm=forms.PatientForm(request.POST,request.FILES,instance=patient)
        if userForm.is_valid() and patientForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()
            patient=patientForm.save(commit=False)
            patient.assignedDoctorId=request.POST.get('assignedDoctorId')
            patient.save()
            return redirect('admin-view-patient')
    return render(request,'hospital/admin_update_patient.html',context=mydict)
```

Delete Patients



```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_patient_from_hospital_view(request,pk):
    patient=models.Patient.objects.get(id=pk)
    user=models.User.objects.get(id=patient.user_id)
    user.delete()
    patient.delete()
    return redirect('admin-view-patient')
```

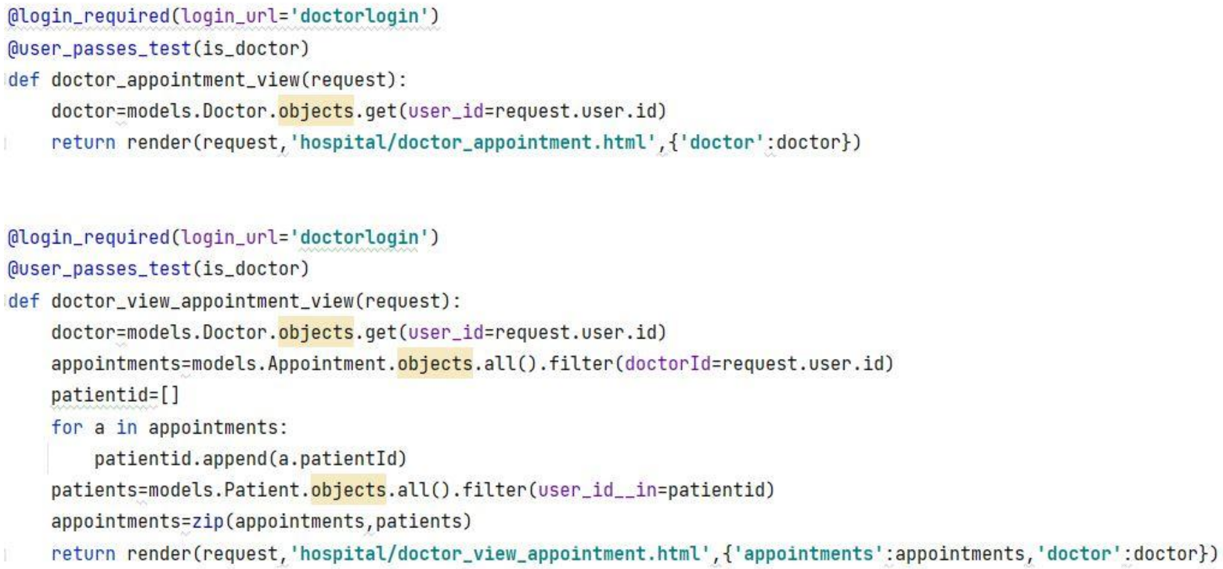
View Patients



```
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_patient_view(request):
    patients=models.Patient.objects.all().filter()
    return render(request,'hospital/admin_view_patient.html',{'patients':patients})
```

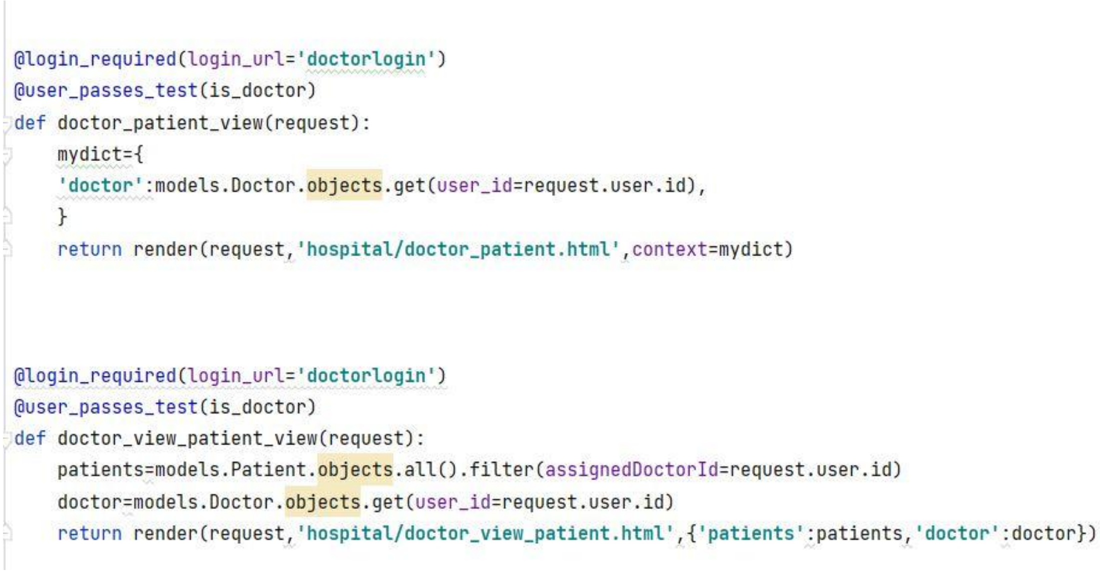
Major Function Prototypes (Doctor)

1. View Appointments



```
@login_required(login_url='doctorlogin')
@user_passes_test(is_doctor)
def doctor_appointment_view(request):
    doctor=models.Doctor.objects.get(user_id=request.user.id)
    return render(request,'hospital/doctor_appointment.html',{'doctor':doctor})


@login_required(login_url='doctorlogin')
@user_passes_test(is_doctor)
def doctor_view_appointment_view(request):
    doctor=models.Doctor.objects.get(user_id=request.user.id)
    appointments=models.Appointment.objects.all().filter(doctorId=request.user.id)
    patientid=[]
    for a in appointments:
        patientid.append(a.patientId)
    patients=models.Patient.objects.all().filter(user_id__in=patientid)
    appointments=zip(appointments,patients)
    return render(request,'hospital/doctor_view_appointment.html',{'appointments':appointments,'doctor':doctor})
```

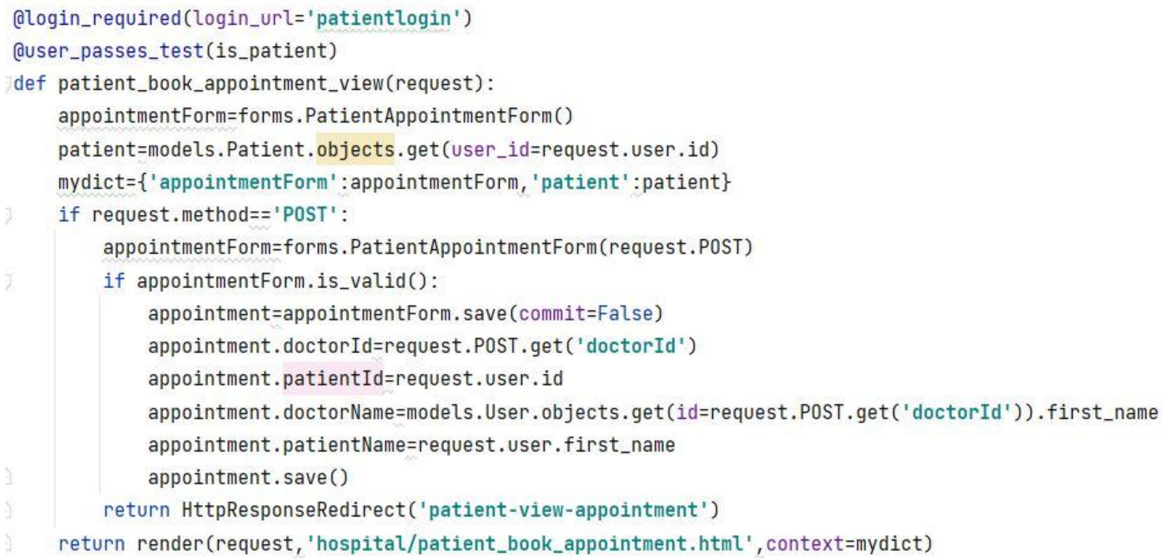
2. View Patients Under them



```
@login_required(login_url='doctorlogin')
@user_passes_test(is_doctor)
def doctor_patient_view(request):
    mydict={
    'doctor':models.Doctor.objects.get(user_id=request.user.id),
    }
    return render(request,'hospital/doctor_patient.html',context=mydict)


@login_required(login_url='doctorlogin')
@user_passes_test(is_doctor)
def doctor_view_patient_view(request):
    patients=models.Patient.objects.all().filter(assignedDoctorId=request.user.id)
    doctor=models.Doctor.objects.get(user_id=request.user.id)
    return render(request,'hospital/doctor_view_patient.html',{'patients':patients,'doctor':doctor})
```

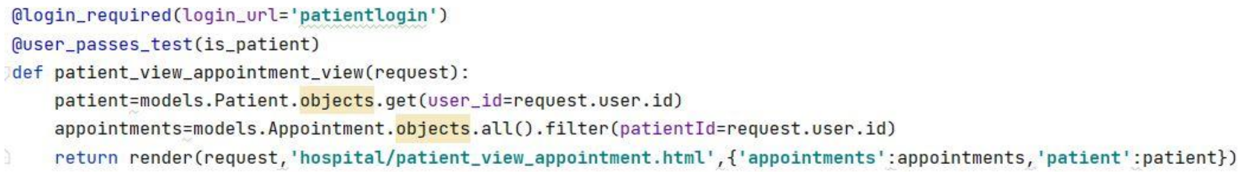
Major Function Prototypes (Patient)

1. Book Appointments



```python
@login_required(login_url='patientlogin')
@user_passes_test(is_patient)
def patient_book_appointment_view(request):
    appointmentForm=forms.PatientAppointmentForm()
    patient=models.Patient.objects.get(user_id=request.user.id)
    mydict={'appointmentForm':appointmentForm,'patient':patient}
    if request.method=='POST':
        appointmentForm=forms.PatientAppointmentForm(request.POST)
        if appointmentForm.is_valid():
            appointment=appointmentForm.save(commit=False)
            appointment.doctorId=request.POST.get('doctorId')
            appointment.patientId=request.user.id
            appointment.doctorName=models.User.objects.get(id=request.POST.get('doctorId')).first_name
            appointment.patientName=request.user.first_name
            appointment.save()
        return HttpResponseRedirect('patient-view-appointment')
    return render(request,'hospital/patient_book_appointment.html',context=mydict)
```

2. View Appointments



```python
@login_required(login_url='patientlogin')
@user_passes_test(is_patient)
def patient_view_appointment_view(request):
    patient=models.Patient.objects.get(user_id=request.user.id)
    appointments=models.Appointment.objects.all().filter(patientId=request.user.id)
    return render(request,'hospital/patient_view_appointment.html',{'appointments':appointments,'patient':patient})
```

Testing

Manual testing was performed in order to find and fix the bugs in the development process.

Testing Method: Manual Testing

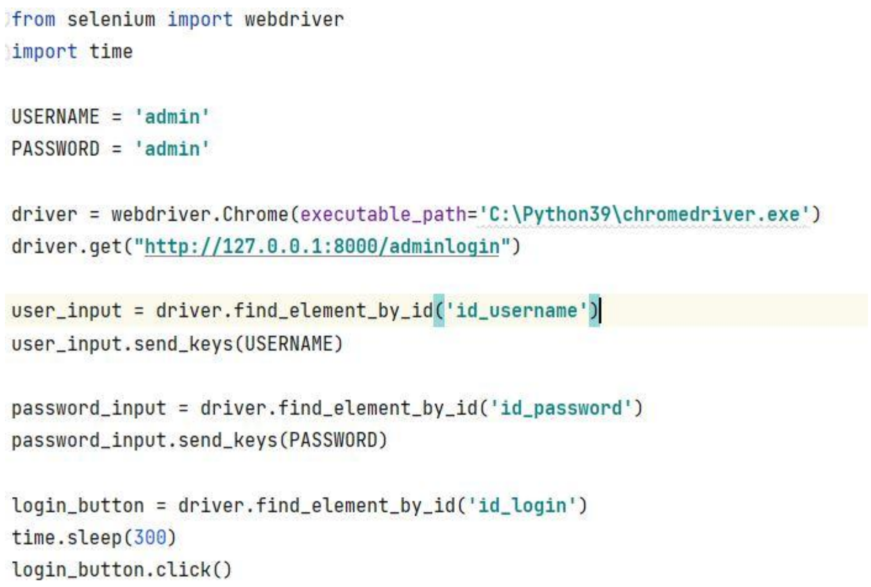
Sr.No.	Test scenario	Expected result	Actual result	Status
1	Login with incorrect credentials	Users should not be able to log in.	User redirected to login page.	Success
2	Login with correct credentials	Users should be able to log in.	User is logged in and shown the home page.	Success
3	Signup with correct details	Users should not be able to sign up with incorrect fields.	Users are not able to sign up with incorrect details.	Success
4	Add doctors	admin should able to add doctors	admin is able to add doctors	Success
5	Remove doctors	admin should able to remove doctors	admin is able to remove doctors	Success
6	Update doctor’s details	admin should able to update details of doctor’s	admin is able to update details of doctor	Success

7	Add patients	admin should able to add patients	admin is able to add patients	Success
8	Remove patients	admin should able to remove patients	admin is able to remove patients	Success
9	Update patients details	admin should able to update details of patients	admin is able to update details of patients	Success
10	Add appointments	patients should able to add appointments	patients is able to add appointments	Success
11	logout	Users should be logged out and restricted from the system until next login.	User is successfully logged out and not able to access the system without signing again.	success

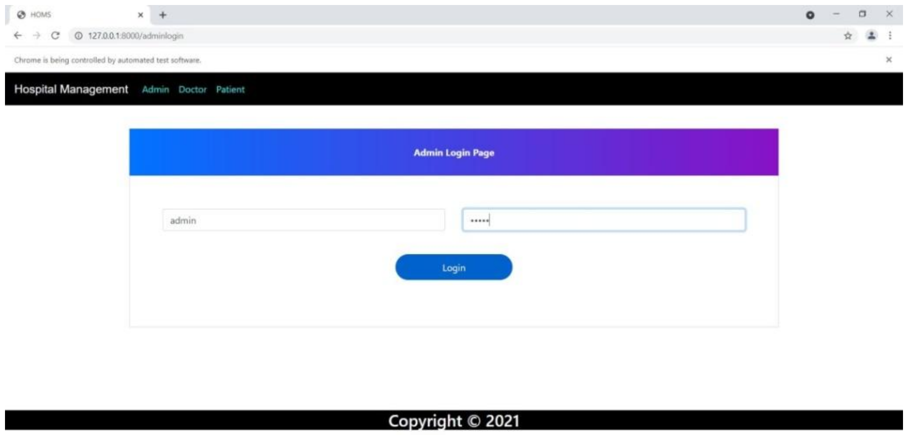
Automation testing was performed using selenium.

Testing Method: Automation Testing

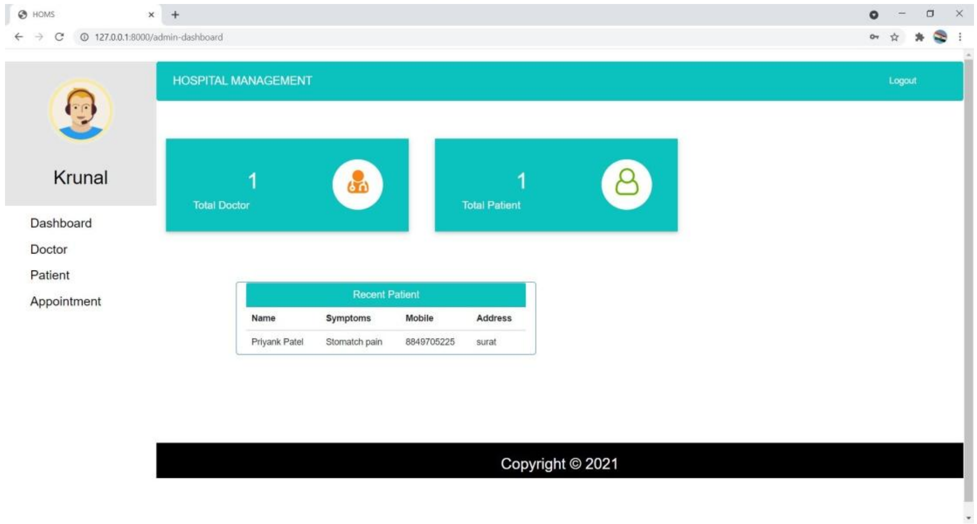
Only login is tested



```
from selenium import webdriver
import time

USERNAME = 'admin'
PASSWORD = 'admin'

driver = webdriver.Chrome(executable_path='C:\Python39\chromedriver.exe')
driver.get("http://127.0.0.1:8000/adminlogin")

user_input = driver.find_element_by_id('id_username')
user_input.send_keys(USERNAME)

password_input = driver.find_element_by_id('id_password')
password_input.send_keys(PASSWORD)

login_button = driver.find_element_by_id('id_login')
time.sleep(300)
login_button.click()
```



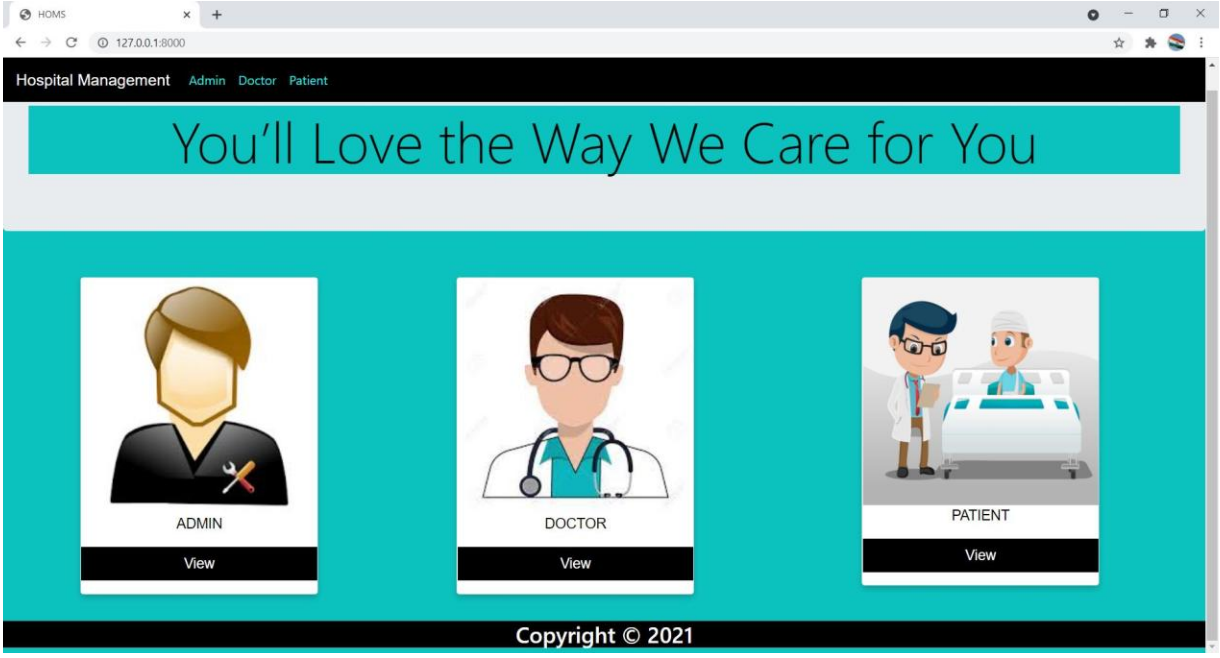
Chrome is being controlled by automated test software.



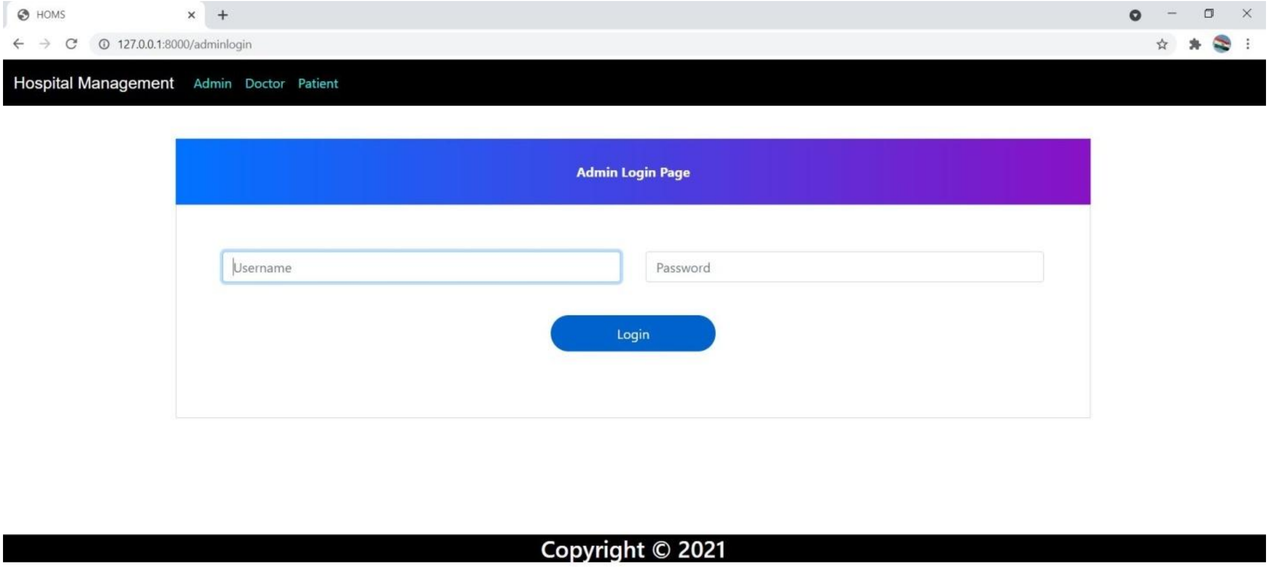
Name	Symptoms	Mobile	Address
Priyank Patel	Stomatch pain	8849705225	surat

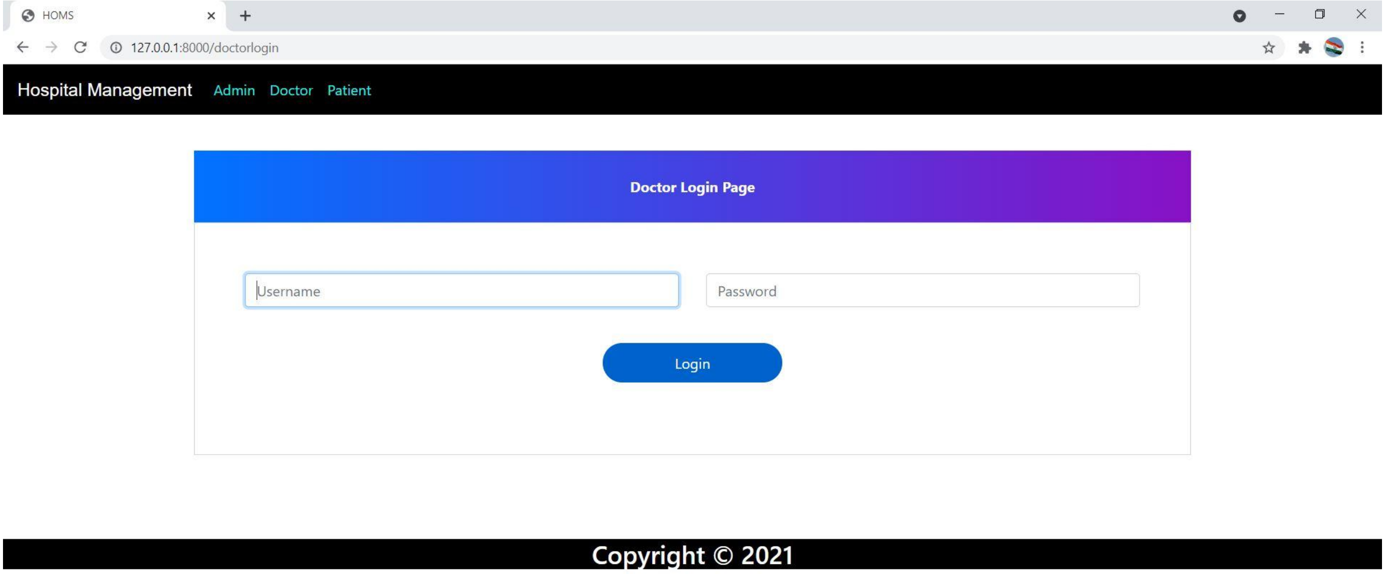
Screenshots(layout)

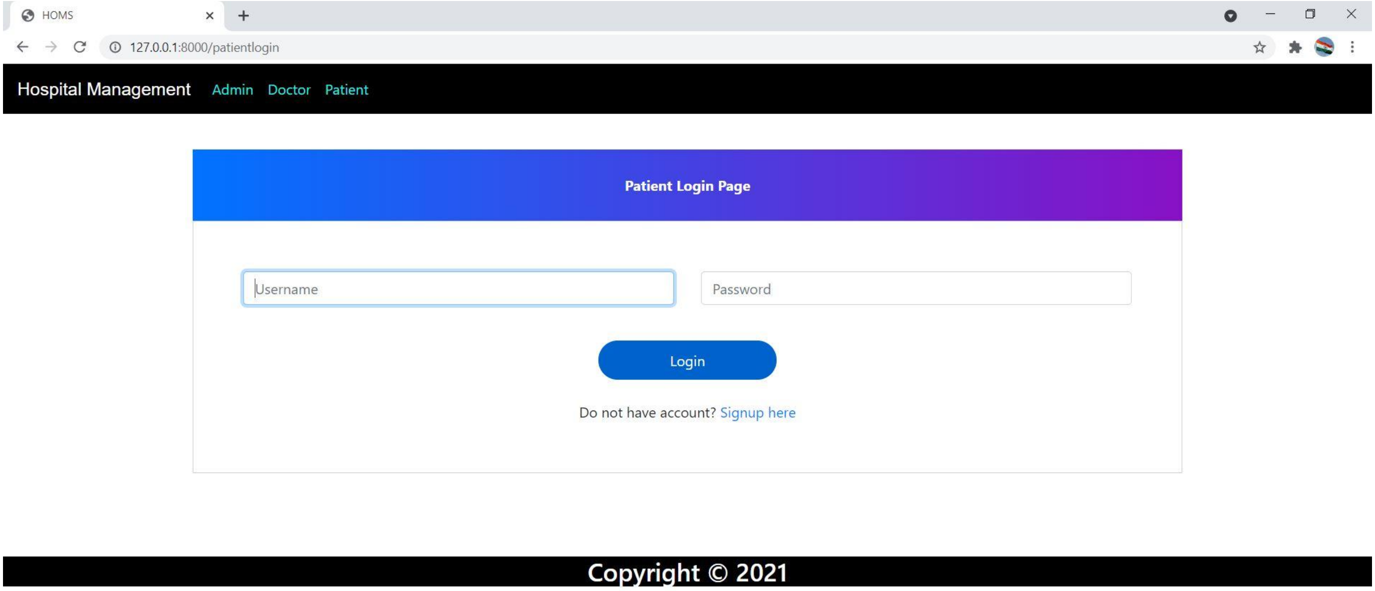
Homepage



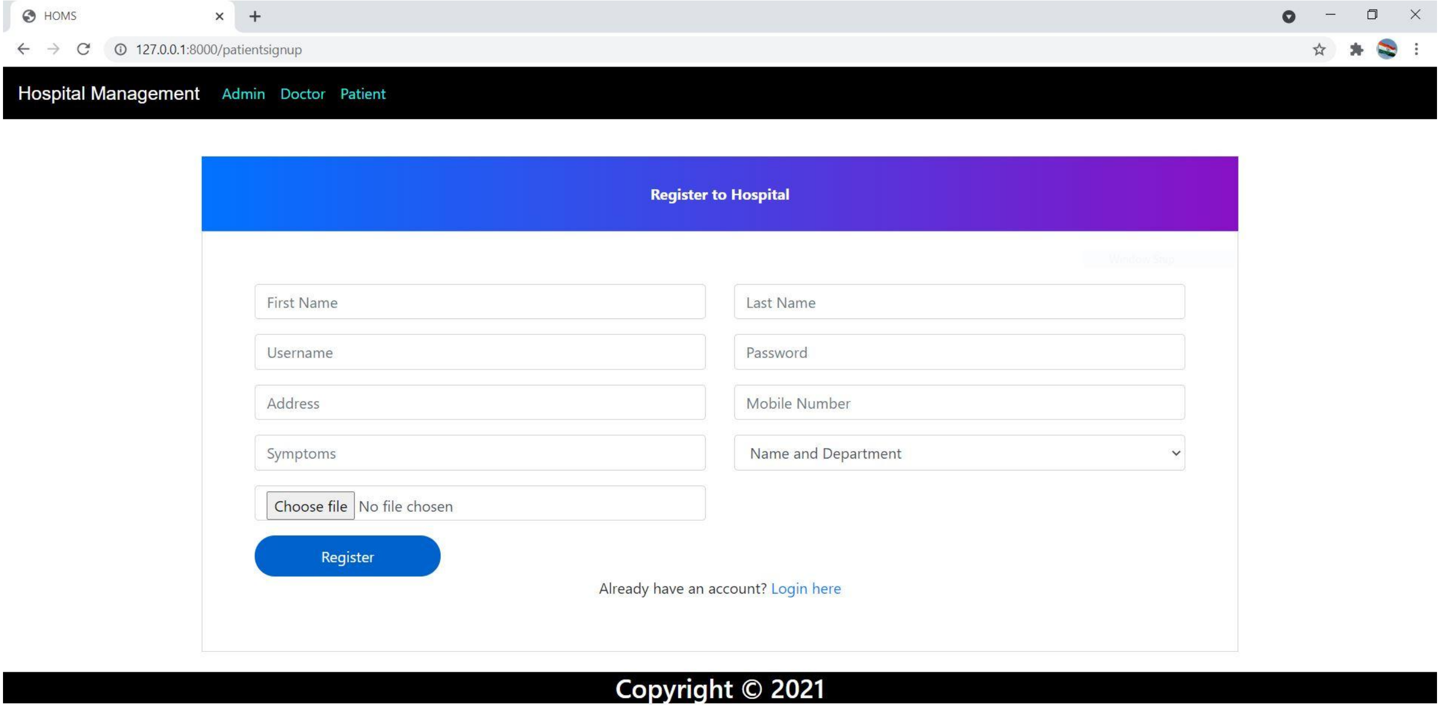
Login



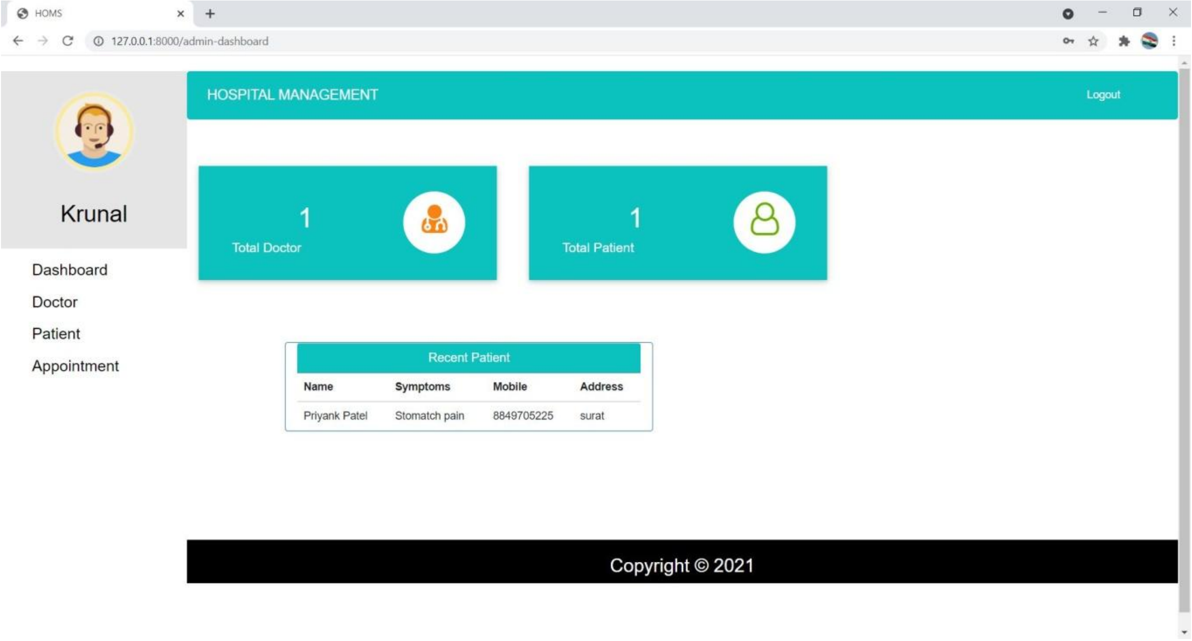

HOMS
127.0.0.1:8000/doctorlogin
Hospital Management
Admin
Doctor
Patient
Doctor Login Page
Username
Password
Login
Copyright © 2021


HOMS
127.0.0.1:8000/patientlogin
Hospital Management
Admin
Doctor
Patient
Patient Login Page
Username
Password
Login
Do not have account? Signup here
Copyright © 2021

Sign up

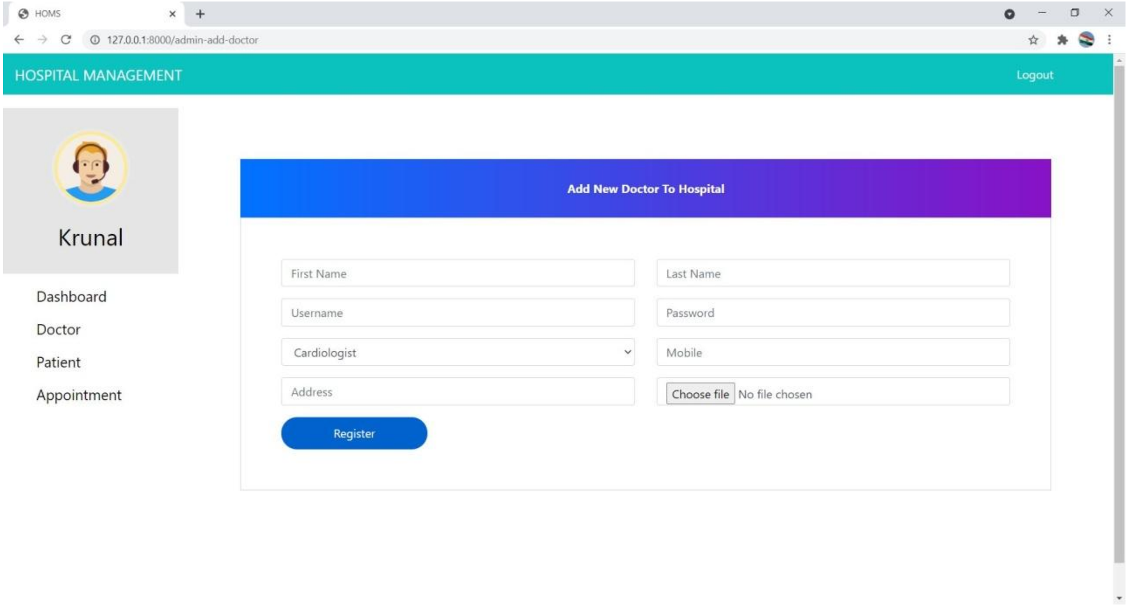


Admin-dashboard



Name	Symptoms	Mobile	Address
Priyank Patel	Stomatch pain	8849705225	surat

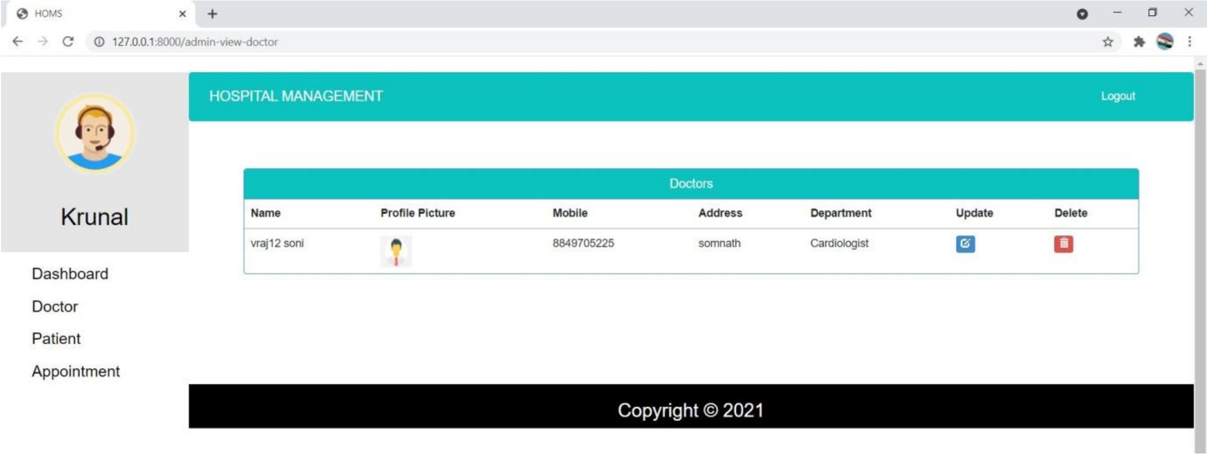
Admin add-doctor



Add New Doctor To Hospital

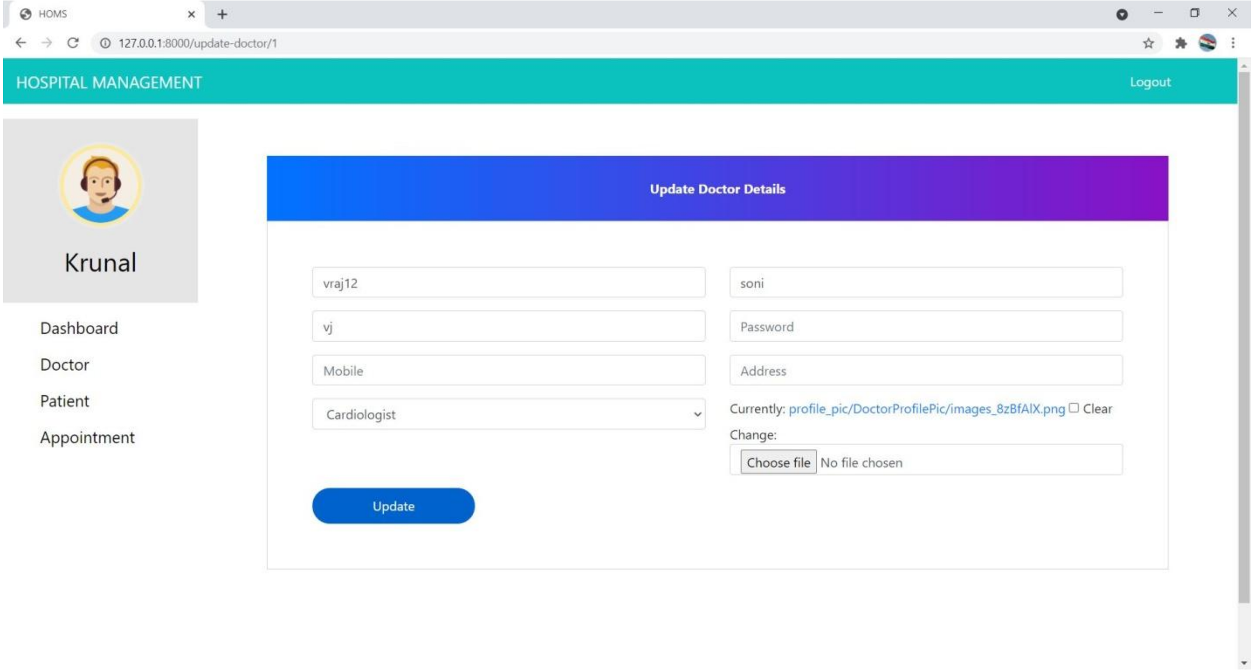
First Name	Last Name
Username	Password
Cardiologist	Mobile
Address	Choose file No file chosen

Admin update-doctor



Doctors

Name	Profile Picture	Mobile	Address	Department	Update	Delete
vraj12 soni		8849705225	somnath	Cardiologist	✎	🗑



Update Doctor Details

vraj12

soni

vj

Password

Mobile

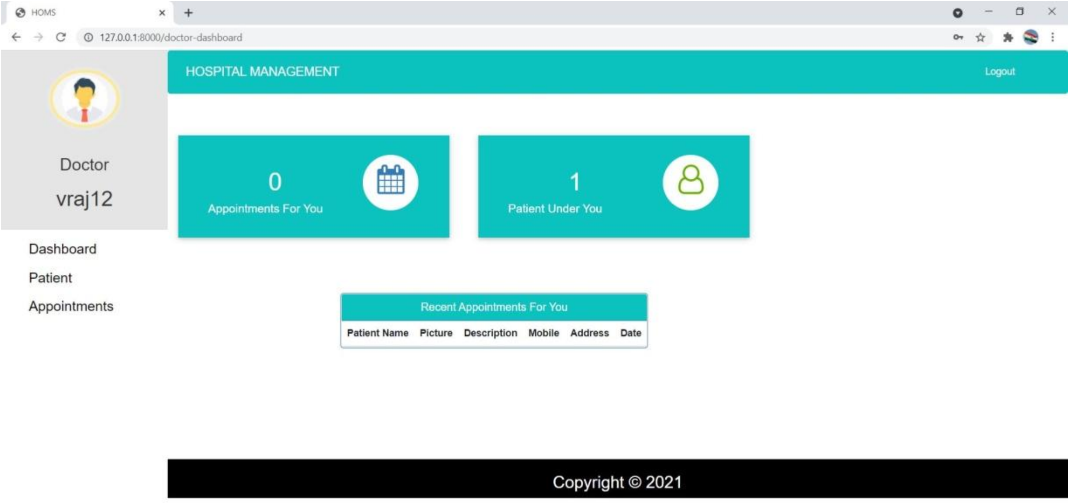
Address

Cardiologist

Currently: profile_pic/DoctorProfilePic/images_8zBfAlX.png □ Clear

Change: Choose file No file chosen

Doctor-dashboard



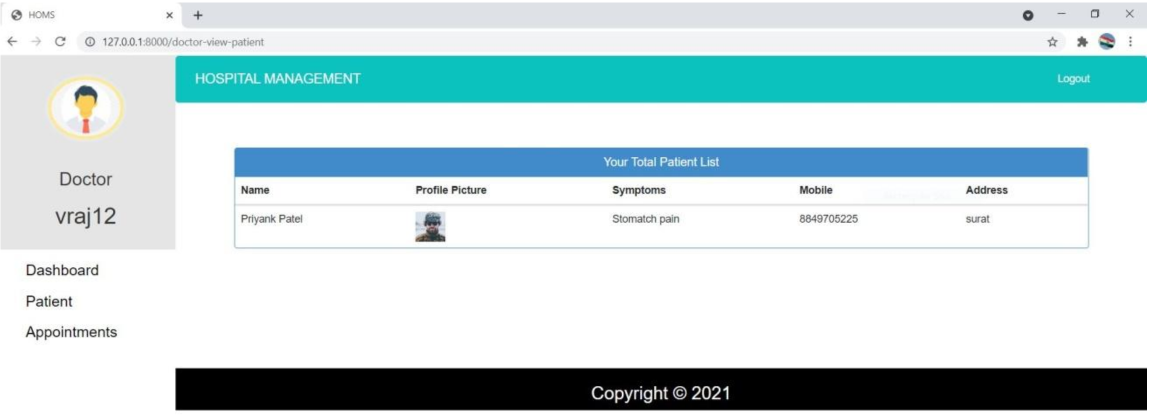
0 Appointments For You

1 Patient Under You

Recent Appointments For You

Patient Name	Picture	Description	Mobile	Address	Date

Doctor view-patient



HOMS

127.0.0.1:8000/doctor-view-patient

HOSPITAL MANAGEMENT

Logout

Doctor

vraj12

Dashboard

Patient

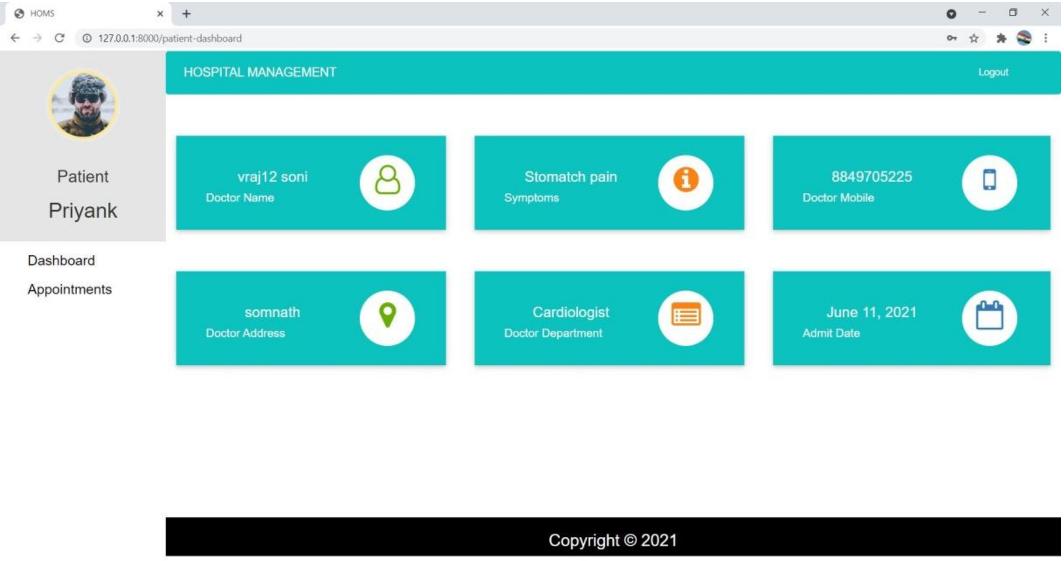
Appointments

Your Total Patient List

Name	Profile Picture	Symptoms	Mobile	Address
Priyank Patel		Stomatch pain	8849705225	surat

Copyright © 2021

Patient-dashboard



HOMS

127.0.0.1:8000/patient-dashboard

HOSPITAL MANAGEMENT

Logout

Patient

Priyank

Dashboard

Appointments

vraj12 soni
Doctor Name

Stomatch pain
Symptoms

8849705225
Doctor Mobile

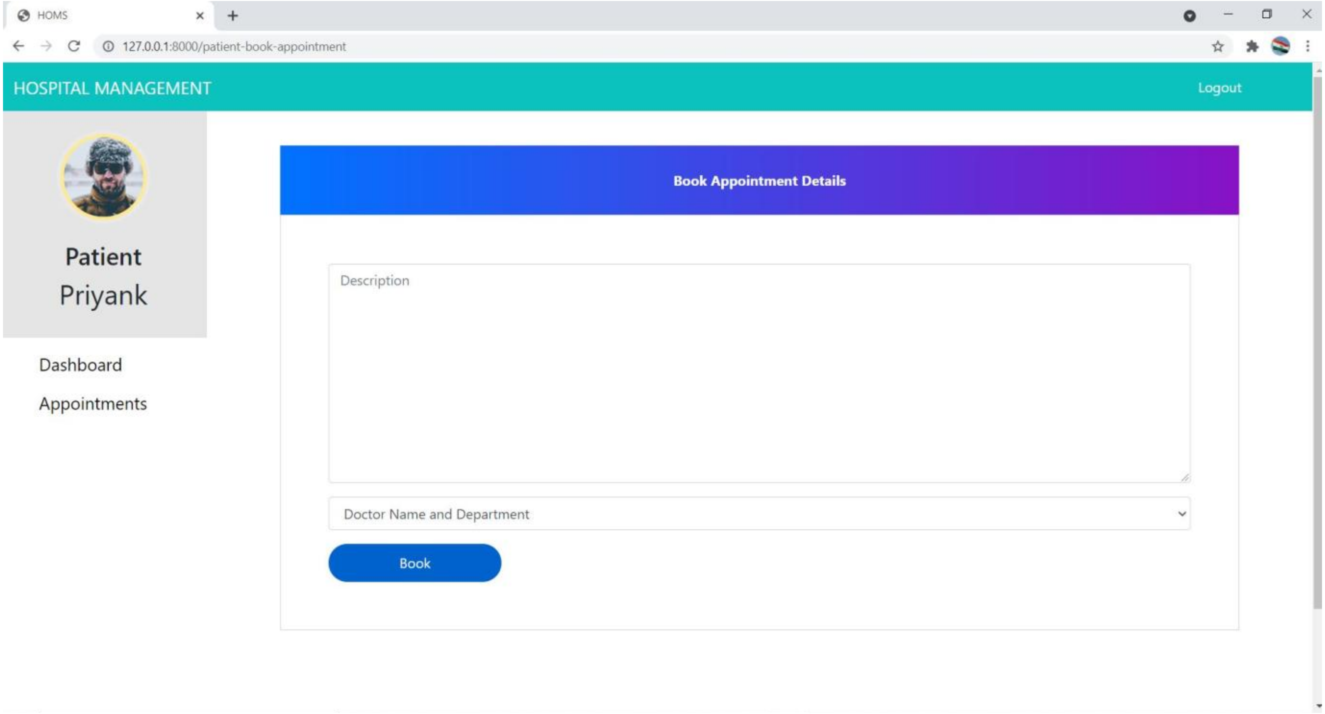
somnath
Doctor Address

Cardiologist
Doctor Department

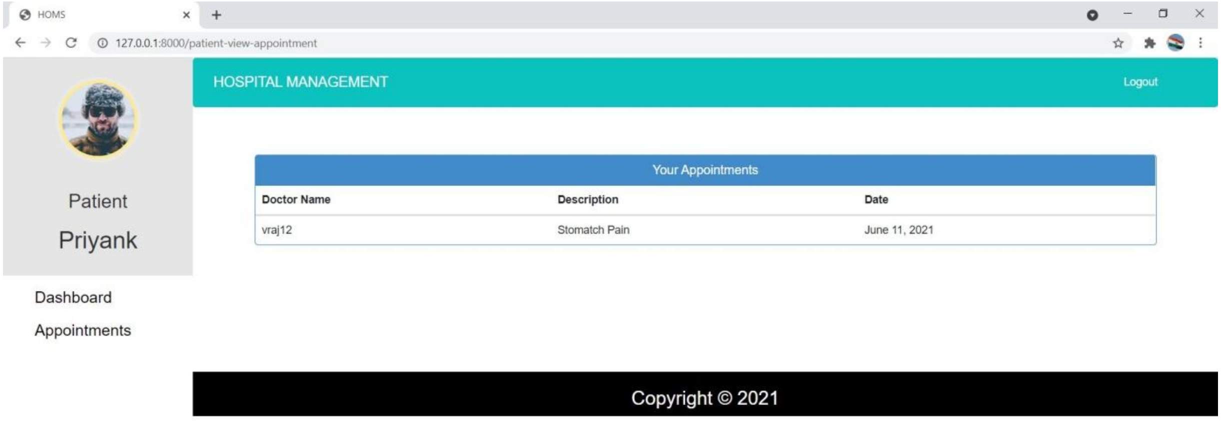
June 11, 2021
Admit Date

Copyright © 2021

Patient Book-appointment



Patient view-appointment



Your Appointments

Doctor Name	Description	Date
vraj12	Stomatch Pain	June 11, 2021

Copyright © 2021

Conclusion

- This is to conclude that the project that we undertook was worked upon with sincere efforts. Most of the requirements have been fulfilled up to the mark and the requirements which have been remaining can be completed with short extension.
- Functionalities that are successfully implemented in the system are:

Functionalities

- Signup, Login, Logout
- Admin can manage/modify doctors and patients.
- Doctors can view appointments.
- Patients can book/view appointments.

Limitations and Future Extensions

We are able to implement most of the functionalities from the "Hospital Management System" functionality module.

We aim to make this product ready to be used in practical use cases.

Limitations:

- Limited number of administration(we have only 1 admin)
- Bill/Report cannot be generated.
- We are lacking a discharge feature in our current version.
- Forgot password option is not given.

Future Extensions:

- Provide online payment options to patients.
- More interactive user interface.
- Search functionality.
- More security options.
- Email verification on signup.
- Forgot password option.
- Bill/Report can be generated.
- A discharge feature will be available.

Bibliography

User interfaces related to this project are taken from the following resources.

- https://www.bootstrapcdn.com/ (styling)
- https://maxcdn.bootstrapcdn.com/font-awesome/ (fonts)